**SOURCE CODE:**

.

**Front-End Implementation**

**App components**

**App.Component.ts**

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'Foodbox-Capstone';

}

* App.Module

import { NgModule } from '@angular/core';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { HomeComponent } from './home/home.component';

import { FoodItemsComponent } from './food-items/food-items.component';

import { AboutusComponent } from './aboutus/aboutus.component';

import { ContactusComponent } from './contactus/contactus.component';

import { LoginComponent } from './login/login.component';

import { RegisterComponent } from './register/register.component';

import { FilterPipe } from 'src/Shared/filter.pipe';

import { HighlightDirective } from 'src/Shared/highlight.pipe';

import { BuyComponent } from './buy/buy.component';

import { CereComponent } from './nonveg/nonveg.component';

import { VegComponent } from './veg/veg.component';

import { SnaksjuiceComponent } from './snaksjuice/snaksjuice.component';

import { HeaderComponent } from './header/header.component';

import { FinalComponent } from './final/final.component';

import { AdminComponent } from './admin/admin.component';

import { AddComponent } from './add/add.component';

import { AddvegComponent } from './addveg/addveg.component';

import { AddcereComponent } from './addnonveg/addnonveg.component';

import { AddsnaksjuiceComponent } from './addsnaksjuice/addsnaksjuice.component';

import { AdpwdComponent } from './adpwd/adpwd.component';

import { AdditemComponent } from './additem/additem.component';

import { AddicComponent } from './addic/addic.component';

import { AddinComponent } from './addin/addin.component';

import { FooterComponent } from './footer/footer.component';

import { PayComponent } from './pay/pay.component';

import { SfooditemsComponent } from './sfooditems/sfooditems.component';

import { NheaderComponent } from './nheader/nheader.component';

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    FoodItemsComponent,

    AboutusComponent,

    ContactusComponent,

    LoginComponent,

    RegisterComponent,

    FilterPipe,

    HighlightDirective,

    BuyComponent,

    nonvegComponent,

    vegComponent,

    SnaksjuiceComponent,

    CoronaComponent,

    HeaderComponent,

    FinalComponent,

    AdminComponent,

    AddComponent,

    AddvegComponent,

    AddnonvegComponent,

    AddSnaksjuiceComponent,

    AdpwdComponent,

    AdditemComponent,

    AddicComponent,

    AddinComponent,

    FooterComponent,

    PayComponent,

    SfooditemsComponent,

    NheaderComponent,

  ],

  imports: [

    BrowserModule,

    FormsModule,

    AppRoutingModule,

    ReactiveFormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

* App.component.spec.ts

import { TestBed } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [

        RouterTestingModule

      ],

      declarations: [

        AppComponent

      ],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have as title 'kitchenstory'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('kitchenstory');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement;

    expect(compiled.querySelector('.content span').textContent).toContain('kitchenstory app is running!');

  });

});

* app.Component.html

<router-outlet></router-outlet>

* app.Component.css

h5

{

    color:goldenrod;

    margin-left: auto;

    margin-right: auto;

}

p{

    font-style:oblique;

    size: 15;

    color:saddlebrown;

}

* **Generating home components**

<app-header></app-header>

  <div id="carouselExampleControls" class="carousel slide" data-ride="carousel">

      <div class="carousel-inner">

        <div class="carousel-item active">

          <img src="assets/harsh.jpg" class="d-block w-100" alt="..." [width]=imgWidth1 [height]=imgHeight1>

        </div>

        <div class="carousel-item">

          <img src="assets/hlp2.jpg" class="d-block w-100" alt="..." [width]=imgWidth1 [height]=imgHeight1>

        </div>

        <div class="carousel-item">

          <a [routerLink]="['/corona']">

          <img src="assets/safety.jpg" class="d-block w-100" alt="..." [width]=imgWidth1 [height]=imgHeight1>

        </a>

        </div>

      </div>

      <a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">

        <span class="carousel-control-prev-icon" aria-hidden="true"></span>

        <span class="sr-only">Previous</span>

      </a>

      <a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">

        <span class="carousel-control-next-icon" aria-hidden="true"></span>

        <span class="sr-only">Next</span>

      </a>

    </div>

  <br>

  <br><br>

  <div class="container">

    <div class="card">

      <div class="card-body">

        <p>Here we provide different range of food items required for your kitchen.</p>

      </div>

    </div>

        <div class="container">

          <div class="jumbotron">

            <img src="assets/harsh1.jpg" width="1000">

            <p>We have best offers and at reasonable prices</p>

            <a class="btn btn-outline-primary" [routerLink]="['/Fooditems']" role="button">Wanna tryout some</a>

          </div>

          </div>

          </div>

<app-footer></app-footer>

  <router-outlet></router-outlet>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {

  let component: HomeComponent;

  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ HomeComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(HomeComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-home',

  templateUrl: './home.component.html',

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  Title="Online Store";

  \_searchTerm:string;

  imgWidth1="600";

  imgHeight1="600";

  imgRadius="15";

  dispck:boolean=false;

  imgDisplay()

    {

        this.dispck=!this.dispck;

    }

  constructor() { }

  ngOnInit(): void {

  }

  Food:any[]=[

    {

    "Item":1,

    "Name":"Broccoli Quinoa Cakes",

    "Cost":550,

    "Rating":5.0,

    "Img":"assets/broccoli-cakes-14.jpg"

    },

    {

      "Item":2,

      "Name":"Roasted Eggplant Lasagna",

      "Cost":500,

      "Rating":4.8,

      "Img":"assets/eggplant.jpg"

      },

      {

        "Item":3,

        "Name":"Indian Shepherd’s Pie",

        "Cost":480,

        "Rating":5.0,

        "Img":"assets/shepherds.jpg"

        },

  ]

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:50px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 900px;

}

p{

    font-style: verdana;

    font-size: 20px;

    font-family: 'Times New Roman', Times, serif;

    font-weight: bold;

}

.nav-link{

    font-style: Arial;

    font-size: 20px;

    font-family: 'Times New Roman', Times, serif;

}

* **Generating Header components**

<nav class=" navbar navbar-expand-lg navbar-light" style="background-color: #40d5e996;">

    <div class="container-fluid">

      <a class="navbar-brand" [routerLink]="['/Fooditems']">

          <img src="/assets/logo1.jpg" height="50px" width="50px">

      </a>

      <a></a>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <ul class="navbar-nav">

        <li class="nav-item active">

          <a class="nav-link" [routerLink]="['/home']">Home</a>

        </li>

        <li class="nav-item active">

          <a class="nav-link" [routerLink]="['/Fooditems']">FoodItems</a>

        </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/login']">Login</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/register']">Register</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link"[routerLink]="['/about']">Aboutus</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/contact']">Contactus</a>

          </li>

        </ul>

        </ul>

        <a></a>

              <a></a>

              <a></a>

      </div>

    </div>

  </nav>

  <br>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HeaderComponent } from './header.component';

describe('HeaderComponent', () => {

  let component: HeaderComponent;

  let fixture: ComponentFixture<HeaderComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ HeaderComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(HeaderComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { foodservice } from '../food-items/food-items.service';

@Component({

  selector: 'app-header',

  templateUrl: './header.component.html',

  styleUrls: ['./header.component.css']

})

export class HeaderComponent implements OnInit {

  constructor(private foo:foodservice) {

    this.foo.cartSubject.subscribe((data)=>{

    this.cartItems=data;

    });

   }

  ngOnInit(): void {

    this.cartItemf()

  }

  cartItems:number=0;

  cartItemf()

  {

    if(localStorage.getItem('cart')!=null)

    {

      var cartcount=JSON.parse(localStorage.getItem('cart'));

      this.cartItems= cartcount.length;

    }

  }

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:50px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

p{

    font-style: verdana;

    font-size: 20px;

}

.nav-link{

    font-style: verdana;

    font-size: 20px;

}

* **Generating fooditems components**

<app-header></app-header>

<div class="container">

<div class="container" style="margin-top:20px;">

  <input class="form-control me-3" type="search" placeholder="Search here" [(ngModel)]="searchterm" aria-label="Search" color="green">

</div>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<div class="container" style="margin-top:20px;">

<div class="row card-group">

<div \*ngFor='let f of searchedItems' class="col-lg-3 col-md-4 col-sm-6 col-xs-12" style="width: 17rem;">

<div class="card mt-4 " >

  <img [src]='f.Img'  class="card-img-top" [width]=imgWidth1 [height]=imgHeight1 >

  <div class="card-body">

    <h5 class="card-title">{{f.Name}}</h5>

    <p class="card-text">Cost:{{f.Cost|currency:"INR"}}<br>Rating:{{f.Rating}}

    </p>

    <div class="content">

       <button type="button" class="btn1" (click)="inc(f)"><i class="fa fa-plus"></i></button>

       <span>{{f.qnt}}</span>

       <button type="button" class="btn1" (click)="dec(f)"><i class="fa fa-minus"></i></button>

      </div>

      <a  class="btn btn-primary"  (click)="od()">Order now!!!</a>

    </div>

  </div>

</div>

</div>

<app-footer>

</app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { FoodItemsComponent } from './food-items.component';

describe('FoodItemsComponent', () => {

  let component: FoodItemsComponent;

  let fixture: ComponentFixture<FoodItemsComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ FoodItemsComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(FoodItemsComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import {efood} from './food-items';

import { foodservice } from './food-items.service';

import { Router } from '@angular/router';

@Component({

  selector: 'app-food-items',

  templateUrl: './food-items.component.html',

  styleUrls: ['./food-items.component.css']

})

export class FoodItemsComponent implements OnInit {

  title = 'angular-text-search-highlight';

  \_searchterm:string;

  imgWidth1="600";

  imgHeight1="300";

  imgRadius="15";

  searchedItems:any[]=[];

    get searchterm():string{

        return this.\_searchterm;

        }

  set searchterm(value:string){

    this.\_searchterm=value;

    this.searchedItems=this.searchterm?this.searchkitchen(this.searchterm):this.Food;

  }

searchkitchen(searchby:string):any[]{

        searchby=searchby.toLocaleLowerCase();

        return this.Food.filter((Food:any)=>Food.Name.toLocaleLowerCase().indexOf(searchby)!==-1);

    }

    constructor(private foo:foodservice,private route:Router) { }

  ngOnInit(): void {

    this.searchedItems=this.Food;

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  itemsCart:any=[];

  add(f)

{

 let cdataNull=localStorage.getItem('cart');

 if(cdataNull==null)

 {

   let sdataget:any=[];

   sdataget.push(f);

   localStorage.setItem('cart',JSON.stringify(sdataget));

 }

 else{

   var id=f.Item;

   let index:number=-1;

   this.itemsCart=JSON.parse(localStorage.getItem('cart'));

   for(let i=0;i<this.itemsCart.length;i++)

   {

     if(parseInt(id) == parseInt(this.itemsCart[i].Item))

     {

         this.itemsCart[i].qnt=f.qnt;

         index=i;

         break;

     }

   }

if(index==-1)

{

  this.itemsCart.push(f);

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

else{

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

 }

 this.cartnumberf();

}

cartNumber:number=0;

  cartnumberf()

  {

      var cartValue=JSON.parse(localStorage.getItem('cart'));

      this.cartNumber= cartValue.length;

      this.foo.cartSubject.next( this.cartNumber);

  }

  od()

  {

    alert("Login First");

    this.route.navigate(['./login']);

  }

  Food:efood[]=[

    {

    "Item":1,

    "Name":"Chicken Biriyani",

    "Cost":299,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/Nv1.png"

    },

    {

      "Item":2,

      "Name":"Chicken Tikka",

      "Cost":350,

      "Rating":4.8,

      "qnt":1,

      "Img":"assets/NV2.png"

      },

      {

        "Item":3,

        "Name":"Chicken Curry",

        "Cost":75,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/NV3.jpg"

        },

        {

          "Item":4,

          "Name":"Veg Biriyani",

          "Cost":100,

          "Rating":5.0,

          "qnt":1,

          "Img":"assets/V1.jpg"

          },

          {

            "Item":5,

            "Name":"Veg Burger",

            "Cost":150,

            "Rating":5.0,

            "qnt":1,

            "Img":"assets/V2.png"

            },

            {

              "Item":6,

              "Name":"Masala Dosa",

              "Cost":250,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/V3.jpg"

              },

               {

              "Item":7,

              "Name":"Samosa",

              "Cost":20,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/S1.jpg"

              },

              {

                "Item":8,

                "Name":"Chips",

                "Cost":15,

                "Rating":5.0,

                "qnt":1,

                "Img":"assets/S2.jpg"

                },

                {

                  "Item":9,

                  "Name":"French Fries",

                  "Cost":120,

                  "Rating":5.0,

                  "qnt":1,

                  "Img":"assets/S3.webp"

                  },

                  {

                    "Item":10,

                    "Name":"Orage Juice",

                    "Cost":12,

                    "Rating":5.0,

                    "qnt":1,

                    "Img":"assets/J1.jpg"

                    },

                    {

                      "Item":11,

                      "Name":"Apple Juice",

                      "Cost":15,

                      "Rating":5.0,

                      "qnt":1,

                      "Img":"assets/J2.jpg"

                      },

                      {

                        "Item":12,

                        "Name":"Pomegranate Juice",

                        "Cost":15,

                        "Rating":5.0,

                        "qnt":1,

                        "Img":"assets/J3.jpg"

                        }

  ]

}

import {Injectable} from "@angular/core";

import {efood} from './food-items';

import {Subject} from 'rxjs';

//import {HttpClient,HttpHeaders} from '@angular/common/http';

@Injectable({

    providedIn:'root'

})

export class foodservice{

  constructor(){}

  // baseurl=global.apiBaseUrl;

   cartSubject=new Subject<any>();

   //openLoginService=new Subject<any>();

 private food:efood[]=[{

    Food:efood[]=[

    {

    "Item":1,

    "Name":"Chicken Biriyani",

    "Cost":299,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/Nv1.png"

    },

    {

      "Item":2,

      "Name":"Chicken Tikka",

      "Cost":350,

      "Rating":4.8,

      "qnt":1,

      "Img":"assets/NV2.png"

      },

      {

        "Item":3,

        "Name":"Chicken Curry",

        "Cost":75,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/NV3.jpg"

        },

        {

          "Item":4,

          "Name":"Veg Biriyani",

          "Cost":100,

          "Rating":5.0,

          "qnt":1,

          "Img":"assets/V1.jpg"

          },

          {

            "Item":5,

            "Name":"Veg Burger",

            "Cost":150,

            "Rating":5.0,

            "qnt":1,

            "Img":"assets/V2.png"

            },

            {

              "Item":6,

              "Name":"Masala Dosa",

              "Cost":250,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/V3.jpg"

              },

               {

              "Item":7,

              "Name":"Samosa",

              "Cost":20,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/S1.jpg"

              },

              {

                "Item":8,

                "Name":"Chips",

                "Cost":15,

                "Rating":5.0,

                "qnt":1,

                "Img":"assets/S2.jpg"

                },

                {

                  "Item":9,

                  "Name":"French Fries",

                  "Cost":120,

                  "Rating":5.0,

                  "qnt":1,

                  "Img":"assets/S3.webp"

                  },

                  {

                    "Item":10,

                    "Name":"Orage Juice",

                    "Cost":12,

                    "Rating":5.0,

                    "qnt":1,

                    "Img":"assets/J1.jpg"

                    },

                    {

                      "Item":11,

                      "Name":"Apple Juice",

                      "Cost":15,

                      "Rating":5.0,

                      "qnt":1,

                      "Img":"assets/J2.jpg"

                      },

                      {

                        "Item":12,

                        "Name":"Pomegranate Juice",

                        "Cost":15,

                        "Rating":5.0,

                        "qnt":1,

                        "Img":"assets/J3.jpg"

                        }

  ]

}

    getFood():efood[]

    {

       return this.food;

    }

    getf(id:number):efood{

     const f1=this.food.find(f1=>f1.Item===id)

     return f1;

    }

}

export interface efood

{

    Item:number,

    Name:string,

    Cost:number,

    Rating:number,

    qnt:number,

    Img:string

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:25px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

p{

    font-style: verdana;

    font-size: 20px;

}

.nav-link{

    font-style: verdana;

    font-size: 20px;

}

.container

{

    color:black;

    font-style: verdana;

    font-size: 20px;

    padding: 10px;

}

* **Generating login components**

<app-header></app-header>

<br />

<body>

<div class="container">

       <p>&nbsp;</p>

       <p>Login Here!!!</p>

<div class="card">

<div class="card-body">

  <form novalidate (ngSubmit)="save(logForm)" #logForm="ngForm">

    <div class="container">

<div class="form-group row mb-6">

  <label class="col-md-2 col-form-label" for="emailId">Email:</label>

  <div class="col-md-8">

    <input type="text" required

                id="emailId"

                name="email"

                [(ngModel)]=login.email

                #email="ngModel"

                pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$"

                [ngClass]="{'is-invalid':(email.touched||email.dirty)&&!email.valid}"/>

                <span class="invalid-feedback">

                  <span \*ngIf=email.errors?.required>

                      Email is required

                  </span>

                  <span \*ngIf=email.errors?.pattern>

                      please enter valid email format

                  </span>

              </span>

          </div>

      </div>

<div class="form-group row mb-2">

  <label class="col-md-2 col-form-label" for="passwordId">Password:</label>

  <div class="col-md-8">

    <input type="password" required id="passwordId" minlength="8" name="password" [(ngModel)]=login.password #passwordVar="ngModel"

    [ngClass]="{'is-invalid':(passwordVar.touched || passwordVar.dirty) && !passwordVar.valid }"/>

  <span class="invalid-feedback">

    <span \*ngIf=passwordVar.errors?.required>

      The password should be entered....

    </span>

    <span \*ngIf=passwordVar.errors?.minlength>

      The password should be atleast 8 characters long....

    </span>

  </span>

  </div>

</div>

<button class="btn btn-success" type="submit" [disabled]="!logForm.valid" >Submit</button>

</div>

</form>

</div>

</div>

<br>

<p> If you are a new user

  <a [routerLink]="['/register']">Regiter Here</a></p>

  <br>

</div>

</body><br><br><br><br>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {

  let component: LoginComponent;

  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ LoginComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(LoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { NgForm} from '@angular/forms';

import { Router } from '@angular/router';

import { Login } from './login';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent  {

  login=new Login();

  constructor(private route:Router){}

  ngonInit():void{}

  save(registerForm:NgForm){

    let uid=this.login.email;

    let pass=this.login.password;

    if(uid=="kmahathi@teksystems.com")

    {

      if(pass=="mahathi123")

      {

        this.route.navigate(['/add']);

      }

      else{

        alert("Admin password incorrect");

      }

    }

    else{

      this.route.navigate(['/sFooditems'])

    }

    console.log(registerForm);

    console.log(JSON.stringify(registerForm.value));

  }

 }

export class Login

{

    constructor(

        public email='',

        public password=''){}

}

export class Customer {

    constructor(

        public firstName = '',

        public lastName = '',

        public email = ''

    ) {

    }

}

* **Generating register components**

<app-header></app-header>

<br />

<body>

<div class="container">

  <p>&nbsp;</p>

      <p  style="color: green;"> Register Here!!!</p>

 <div class="card">

<div class="card-body">

  <form novalidate (ngSubmit)="save(regForm)" #regForm="ngForm">

    <div class="form-group row mb-2">

      <label class="col-md-2 col-form-label" for="firstnameId">FirstName:</label>

      <div class="col-md-8">

        <input type="text" required id="firstnameId" minlength="4" name="firstName" [(ngModel)]=register.firstName #firstNameVar="ngModel"

        [ngClass]="{'is-invalid':(firstNameVar.touched || firstNameVar.dirty) && !firstNameVar.valid }"/>

      <span class="invalid-feedback">

        <span \*ngIf=firstNameVar.errors?.required>

          The firstName should be entered....

        </span>

        <span \*ngIf=firstNameVar.errors?.minlength>

          The firstName should be atleast 4 characters long....

        </span>

      </span>

      </div>

    </div>

<div class="form-group row mb-2">

  <label class="col-md-2 col-form-label" for="firstnameId">LastName:</label>

  <div class="col-md-8">

    <input type="text" required id="lastnameId" minlength="4" name="lastName" [(ngModel)]=register.lastName #lastNameVar="ngModel"

    [ngClass]="{'is-invalid':(lastNameVar.touched || lastNameVar.dirty) && !lastNameVar.valid }"/>

  <span class="invalid-feedback">

    <span \*ngIf=lastNameVar.errors?.required>

      The LastName should be entered....

    </span>

    <span \*ngIf=lastNameVar.errors?.minlength>

      The lastName should be atleast 4 characters long....

    </span>

  </span>

  </div>

</div>

<div class="form-group row mb-2">

  <label class="col-md-2 col-form-label" for="emailId">Email:</label>

  <div class="col-md-8">

    <input type="text" required

                id="emailId"

                name="email"

                [(ngModel)]=register.email

                #email="ngModel"

                pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$"

                [ngClass]="{'is-invalid':(email.touched||email.dirty)&&!email.valid}"/>

                <span class="invalid-feedback">

                  <span \*ngIf=email.errors?.required>

                      Email is required

                  </span>

                  <span \*ngIf=email.errors?.pattern>

                      please enter valid email format

                  </span>

              </span>

          </div>

      </div>

<div class="form-group row mb-2">

  <label class="col-md-2 col-form-label" for="passwordId">SetPassword:</label>

  <div class="col-md-8">

    <input type="password" required id="passwordId" minlength="8" name="password" [(ngModel)]=register.password #passwordVar="ngModel"

    [ngClass]="{'is-invalid':(passwordVar.touched || passwordVar.dirty) && !passwordVar.valid }"/>

  <span class="invalid-feedback">

    <span \*ngIf=passwordVar.errors?.required>

      The password should be entered....

    </span>

    <span \*ngIf=passwordVar.errors?.minlength>

      The password should be atleast 8 characters long....

    </span>

  </span>

  </div>

</div>

<button class="btn btn-success" type="submit" [disabled]="!regForm.valid" [routerLink]="['/login']">Submit</button>

</form>

</div>

</div>

<br>

<p > If Already registered ??

  <a [routerLink]="['/login']">Get Login</a></p>

  <br>

</div>

</body><br><br><br><br>

 <app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { RegisterComponent } from './register.component';

describe('RegisterComponent', () => {

  let component: RegisterComponent;

  let fixture: ComponentFixture<RegisterComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ RegisterComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(RegisterComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { stringify } from '@angular/compiler/src/util';

import { Component, OnInit } from '@angular/core';

import { NgForm } from '@angular/forms';

import { Register } from './register';

@Component({

  selector: 'app-register',

  templateUrl: './register.component.html',

  styleUrls: ['./register.component.css']

})

export class RegisterComponent  {

 register=new Register();

 save(registerForm:NgForm){

   console.log(registerForm);

   console.log(JSON.stringify(registerForm.value));

 }

}

export class Register

{

    constructor(

        public firstName="",

        public lastName="",

        public email='',

        public password=''){}

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:50px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 900px;

}

p{

    font-style: verdana;

    font-size: 25px;

    font-family: 'Times New Roman', Times, serif;

    font-weight: bold;

}

.nav-link{

    font-style: verdana;

    font-size: 20px;

}

/\* body

{

    background-image: url("../../assets/logo.jpg");

} \*/

* **Generating contactus components**

<app-header></app-header>

<div class="container">

<form>

    <div class="form-group">

      <label for="exampleFormControlInput1">Name</label>

      <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="Name">

    </div>

    <div class="form-group">

        <label for="exampleFormControlInput1">Email address</label>

        <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name@example.com">

      </div>

    <div class="form-group">

      <label for="exampleFormControlTextarea1">Enter issue</label>

      <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>

    </div>

    <button [routerLink]="['/home']" class="btn btn-primary" type="submit">Submit</button>

  </form><br><br><br><br><br><br>

</div>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ContactusComponent } from './contactus.component';

describe('ContactusComponent', () => {

  let component: ContactusComponent;

  let fixture: ComponentFixture<ContactusComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ ContactusComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(ContactusComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-contactus',

  templateUrl: './contactus.component.html',

  styleUrls: ['./contactus.component.css']

})

export class ContactusComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

* **Generating About us components**

<app-header></app-header>

<div>

    <p>

        This website is designed in order to provide various basic food items which are required in kitchen.The basic goal is to provide the items with best quality.

        Quality is our first priority.

    </p>

    <p>It is basically an idea of providing  all basic needs in kitchen.All the basic food items in the kitchen are available so that people can get their required items easily  </p>

    <div class="centered">

</div>

</div><br><br><br><br><br><br><br><br>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AboutusComponent } from './aboutus.component';

describe('AboutusComponent', () => {

  let component: AboutusComponent;

  let fixture: ComponentFixture<AboutusComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AboutusComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AboutusComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-aboutus',

  templateUrl: './aboutus.component.html',

  styleUrls: ['./aboutus.component.css']

})

export class AboutusComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

.right

{   top:auto;

    right: 16px;

    color: cyan;

}

.centered {

    position: absolute;

    top: 75%;

    left: 55%;

    transform: translate(-50%, -50%);

  }

  p{

      font-style:normal ;

      font-family: 'Times New Roman', Times, serif;

      font-size: 25px;

      color: 000;

  }

* **Generating footer components**

<footer class="bg-light text-center text-white">

  <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.89);">

    <!-- Copyright -->

      © 2022 Copyright:

      <a class="text-white" href="https://mdbootstrap.com/">Harsha M</a>

    </div>

    <!-- Copyright -->

  </footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { FooterComponent } from './footer.component';

describe('FooterComponent', () => {

  let component: FooterComponent;

  let fixture: ComponentFixture<FooterComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ FooterComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(FooterComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-footer',

  templateUrl: './footer.component.html',

  styleUrls: ['./footer.component.css']

})

export class FooterComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

* **Generating Veg components**

<app-nheader></app-nheader>

<div class="container">

<div class="d-grid gap-2 d-md-flex ">

  &nbsp;&nbsp;&nbsp;

<a [routerLink]="['/fruit']" class="nav" role="button">veg</a>&nbsp;&nbsp;

<a [routerLink]="['/cere']" class="nav" role="button">non-veg</a>&nbsp;&nbsp;

<a [routerLink]="['/nut']" class="nav" role="button">Snaksjuice</a><a></a>&nbsp;&nbsp;

</div>

</div>

<div class="container" style="margin-top:20px;">

  <input class="form-control me-3" type="search" placeholder="Search here" [(ngModel)]="searchterm" aria-label="Search" color="green">

</div>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<div class="container" style="margin-top:20px;">

<div class="row card-group">

<div \*ngFor='let f of searchedItems' class="col-lg-3 col-md-4 col-sm-6 col-xs-12" style="width: 17rem;">

<div class="card mt-4 " >

  <img [src]='f.Img'  class="card-img-top" [width]=imgWidth1 [height]=imgHeight1 >

  <div class="card-body">

    <h5 class="card-title">{{f.Name}}</h5>

    <p class="card-text">Cost:{{f.Cost|currency:"INR"}}<br>Rating:{{f.Rating}}

    </p>

    <div class="content">

       <button type="button" class="btn1" (click)="inc(f)"><i class="fa fa-plus"></i></button>

       <span>{{f.qnt}}</span>

       <button type="button" class="btn1" (click)="dec(f)"><i class="fa fa-minus"></i></button>

      </div>

      <a  class="btn btn-primary"  (click)="add(f)">Add to Cart</a>

    </div>

  </div>

</div>

</div>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { FruitComponent } from './fruit.component';

describe('FruitComponent', () => {

  let component: FruitComponent;

  let fixture: ComponentFixture<FruitComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ FruitComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(FruitComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { efood } from '../food-items/food-items';

import { foodservice } from 'src/app/food-items/food-items.service';

@Component({

  selector: 'app-veg',

  templateUrl: './veg.component.html',

  styleUrls: ['./veg.component.css']

})

export class vegComponent implements OnInit {

  \_searchterm:string;

  imgWidth1="600";

  imgHeight1="300";

  imgRadius="15";

  searchedItems:any[]=[];

    get searchterm():string{

        return this.\_searchterm;

        }

  set searchterm(value:string){

    this.\_searchterm=value;

    this.searchedItems=this.searchterm?this.searchkitchen(this.searchterm):this.fruit;

  }

searchkitchen(searchby:string):any[]{

        searchby=searchby.toLocaleLowerCase();

        return this.fruit.filter((Food:any)=>Food.Name.toLocaleLowerCase().indexOf(searchby)!==-1);

    }

    constructor(private foo:foodservice) { }

  ngOnInit(): void {

    this.searchedItems=this.fruit;

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  itemsCart:any=[];

  add(f)

{

 let cdataNull=localStorage.getItem('cart');

 if(cdataNull==null)

 {

   let sdataget:any=[];

   sdataget.push(f);

   localStorage.setItem('cart',JSON.stringify(sdataget));

 }

 else{

   var id=f.Item;

   let index:number=-1;

   this.itemsCart=JSON.parse(localStorage.getItem('cart'));

   for(let i=0;i<this.itemsCart.length;i++)

   {

     if(parseInt(id) == parseInt(this.itemsCart[i].Item))

     {

         this.itemsCart[i].qnt=f.qnt;

         index=i;

         break;

     }

   }

if(index==-1)

{

  this.itemsCart.push(f);

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

else{

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

 }

 this.cartnumberf();

}

cartNumber:number=0;

  cartnumberf()

  {

      var cartValue=JSON.parse(localStorage.getItem('cart'));

      this.cartNumber= cartValue.length;

      this.foo.cartSubject.next( this.cartNumber);

  }

  fruit:efood[]=[

    {

      "Item":4,

      "Name":"Oranges",

      "Cost":100,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/oranges.jpg"

      },

      {

        "Item":5,

        "Name":"Pomegranate",

        "Cost":150,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/pome.jpg"

        },

        {

          "Item":6,

          "Name":"Apples",

          "Cost":250,

          "Rating":5.0,

          "qnt":1,

          "Img":"assets/apples.jpg"

          }

        ]

}

import {Injectable} from "@angular/core";

import { efood } from '../food-items/food-items';

import {Subject} from 'rxjs';

//import {HttpClient,HttpHeaders} from '@angular/common/http';

@Injectable({

    providedIn:'root'

})

export class vegservice{

  constructor(){}

  // baseurl=global.apiBaseUrl;

   cartSubject=new Subject<any>();

   //openLoginService=new Subject<any>();

 private fruit:efood[]=[{

    "Item":4,

  "Name":"Veg Biriyani",

  "Cost":100,

  "Rating":5.0,

  "qnt":1,

  "Img":"assets/V1.jpg"

  },

  {

    "Item":5,

    "Name":"Veg Burger",

    "Cost":40,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/V2.png"

    },

    {

      "Item":6,

      "Name":"Masala Dosa",

      "Cost":55,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/V3.jpg"

      },

      {

        "Item":21,

        "Name":"Veg Fried Rice",

        "Cost":80,

        "Rating":4.5,

        "qnt":1,

        "Img":"assets/V4.jpg"

        }

      ]

      getFruit():efood[]

      {

         return this.veg;

      }

      getf(id:number):efood{

       const f1=this.veg.find(f1=>f1.Item===id)

       return f1;

      }

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:25px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

.container

{

    color:black;

    font-style: verdana;

    font-size: 20px;

    padding: 10px;

}

* **Generating SnaksJuice components**

<app-nheader></app-nheader>

<div class="container">

  <div class="d-grid gap-2 d-md-flex ">

    &nbsp;&nbsp;&nbsp;

  <a [routerLink]="['/veg']" class="nav" role="button">Veg </a>&nbsp;&nbsp;

  <a [routerLink]="['/nonveg']" class="nav" role="button">Non-Veg</a>&nbsp;&nbsp;

  <a [routerLink]="['/snaksjuice']" class="nav" role="button">Snaksjuice</a><a></a>&nbsp;&nbsp;

  </div>

  </div>

<div class="container" style="margin-top:20px;">

  <input class="form-control me-3" type="search" placeholder="Search here" [(ngModel)]="searchterm" aria-label="Search" color="green">

</div>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<div class="container" style="margin-top:20px;">

<div class="row card-group">

<div \*ngFor='let f of searchedItems' class="col-lg-3 col-md-4 col-sm-6 col-xs-12" style="width: 17rem;">

<div class="card mt-4 " >

  <img [src]='f.Img'  class="card-img-top" [width]=imgWidth1 [height]=imgHeight1 >

  <div class="card-body">

    <h5 class="card-title">{{f.Name}}</h5>

    <p class="card-text">Cost:{{f.Cost|currency:"INR"}}<br>Rating:{{f.Rating}}

    </p>

    <div class="content">

       <button type="button" class="btn1" (click)="inc(f)"><i class="fa fa-plus"></i></button>

       <span>{{f.qnt}}</span>

       <button type="button" class="btn1" (click)="dec(f)"><i class="fa fa-minus"></i></button>

      </div>

      <a  class="btn btn-primary"  (click)="add(f)">Add to Cart</a>

    </div>

  </div>

</div>

</div>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { SnaksjuiceComponent } from './snaksjuice.component';

describe('SnaksjuiceComponent', () => {

  let component: SnaksjuiceComponent;

  let fixture: ComponentFixture<NutComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ SnaksjuiceComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(SnaksjuiceComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { efood } from '../food-items/food-items';

import { foodservice } from 'src/app/food-items/food-items.service';

@Component({

  selector: 'app-Snaksjuice',

  templateUrl: './Snakjuice.component.html',

  styleUrls: ['./Snaksjuice.component.css']

})

export class SnaksjuiceComponent implements OnInit {

  \_searchterm:string;

  imgWidth1="600";

  imgHeight1="300";

  imgRadius="15";

  searchedItems:any[]=[];

    get searchterm():string{

        return this.\_searchterm;

        }

  set searchterm(value:string){

    this.\_searchterm=value;

    this.searchedItems=this.searchterm?this.searchkitchen(this.searchterm):this.nut;

  }

searchkitchen(searchby:string):any[]{

        searchby=searchby.toLocaleLowerCase();

        return this.nut.filter((Food:any)=>Food.Name.toLocaleLowerCase().indexOf(searchby)!==-1);

    }

    constructor(private foo:foodservice) { }

  ngOnInit(): void {

    this.searchedItems=this.nut;

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  itemsCart:any=[];

  add(f)

{

 let cdataNull=localStorage.getItem('cart');

 if(cdataNull==null)

 {

   let sdataget:any=[];

   sdataget.push(f);

   localStorage.setItem('cart',JSON.stringify(sdataget));

 }

 else{

   var id=f.Item;

   let index:number=-1;

   this.itemsCart=JSON.parse(localStorage.getItem('cart'));

   for(let i=0;i<this.itemsCart.length;i++)

   {

     if(parseInt(id) == parseInt(this.itemsCart[i].Item))

     {

         this.itemsCart[i].qnt=f.qnt;

         index=i;

         break;

     }

   }

if(index==-1)

{

  this.itemsCart.push(f);

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

else{

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

 }

 this.cartnumberf();

}

cartNumber:number=0;

  cartnumberf()

  {

      var cartValue=JSON.parse(localStorage.getItem('cart'));

      this.cartNumber= cartValue.length;

      this.foo.cartSubject.next( this.cartNumber);

  }

  nut:efood[]=[

"Item":7,

  "Name":"Samosa",

  "Cost":20,

  "Rating":5.0,

  "qnt":1,

  "Img":"assets/S1.jpg"

  },

  {

    "Item":8,

    "Name":"Chips",

    "Cost":15,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/S2.jpg"

    },

    {

      "Item":9,

      "Name":"French Fries",

      "Cost":120,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/S3.webp"

      },

      {

        "Item":10,

        "Name":"Orage Juice",

        "Cost":12,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/J1.jpg"

        },

        {

          "Item":11,

          "Name":"Apple Juice",

          "Cost":15,

          "Rating":5.0,

          "qnt":1,

          "Img":"assets/J2.jpg"

          },

          {

            "Item":12,

            "Name":"Pomegranate Juice",

            "Cost":15,

            "Rating":5.0,

            "qnt":1,

            "Img":"assets/J3.jpg"

            }

]

}

import {Injectable} from "@angular/core";

import { efood } from '../food-items/food-items';

import {Subject} from 'rxjs';

//import {HttpClient,HttpHeaders} from '@angular/common/http';

@Injectable({

    providedIn:'root'

})

export class Snaksjuiceservice{

  constructor(){}

  // baseurl=global.apiBaseUrl;

   cartSubject=new Subject<any>();

   //openLoginService=new Subject<any>();

 private cere:efood[]=[{

    "Item":7,

    "Name":"Almonds",

    "Cost":700,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/almond.jpg"

    },

    {

      "Item":8,

      "Name":"Cashew",

      "Cost":500,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/cashew.jpg"

      },

      {

        "Item":9,

        "Name":"Groundnuts",

        "Cost":120,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/ground.jpg"

        },

]

getnut():efood[]

{

   return this.cere;

}

getf(id:number):efood{

 const f1=this.cere.find(f1=>f1.Item===id)

 return f1;

}

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:25px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

.container

{

    color:black;

    font-style: verdana;

    font-size: 20px;

    padding: 10px;

}

* **Generating nheader components**

<nav class=" navbar navbar-expand-lg navbar-light" style="background-color: #e3f2fd;">

    <div class="container-fluid">

      <a class="navbar-brand" [routerLink]="['/Fooditems']">

          <img src="/assets/logo1.jpg" height="50px" width="50px">

      </a>

      <a></a>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <ul class="navbar-nav">

        <li class="nav-item active">

          <a class="nav-link" [routerLink]="['/home']">Home</a>

        </li>

        <li class="nav-item active">

          <a class="nav-link" [routerLink]="['/sFooditems']">FoodItems</a>

        </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/login']">Login</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/register']">Register</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link"[routerLink]="['/about']">Aboutus</a>

          </li>

          <li class="nav-item active">

            <a class="nav-link" [routerLink]="['/contact']">Contactus</a>

          </li>

        </ul>

        </ul>

        <a></a>

              <a></a>

              <a></a>

        <ul class="navbar-nav ml-auto">

            <li class="nav-item">

                <div class="col-md-12 text-right">

                  <div class="cart\_icon" [routerLink]="['/buy']">

                    <i class="fa fa-shopping-cart"></i>

                    <div class="cart\_value">

                      {{cartItems}}

                    </div>

              </div>

              </div>

              </li>

              <li class="nav-item active">

                <a class="nav-link" [routerLink]="['/home']">Logout</a>

              </li>

              <a></a>

              <a></a>

              <a></a>

            </ul>

      </div>

    </div>

  </nav>

  <br>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NheaderComponent } from './nheader.component';

describe('NheaderComponent', () => {

  let component: NheaderComponent;

  let fixture: ComponentFixture<NheaderComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ NheaderComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(NheaderComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { foodservice } from '../food-items/food-items.service';

@Component({

  selector: 'app-nheader',

  templateUrl: './nheader.component.html',

  styleUrls: ['./nheader.component.css']

})

export class NheaderComponent implements OnInit {

  constructor(private foo:foodservice) {

    this.foo.cartSubject.subscribe((data)=>{

    this.cartItems=data;

    });

   }

  ngOnInit(): void {

    this.cartItemf()

  }

  cartItems:number=0;

  cartItemf()

  {

    if(localStorage.getItem('cart')!=null)

    {

      var cartcount=JSON.parse(localStorage.getItem('cart'));

      this.cartItems= cartcount.length;

    }

  }

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:50px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

p{

    font-style: verdana;

    font-size: 20px;

}

.nav-link{

    font-style: verdana;

    font-size: 20px;

}

* **Generating nonveg components**

<app-nheader></app-nheader>

<div class="container">

  <div class="d-grid gap-2 d-md-flex ">

    &nbsp;&nbsp;&nbsp;

  <a [routerLink]="['/fruit']" class="nav" role="button">veg </a>&nbsp;&nbsp;

  <a [routerLink]="['/cere']" class="nav" role="button">nonveg</a>&nbsp;&nbsp;

  <a [routerLink]="['/nut']" class="nav" role="button">Snaksjuice</a><a></a>&nbsp;&nbsp;

  </div>

  </div>

<div class="container" style="margin-top:20px;">

  <input class="form-control me-3" type="search" placeholder="Search here" [(ngModel)]="searchterm" aria-label="Search" color="green">

</div>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<div class="container" style="margin-top:20px;">

<div class="row card-group">

<div \*ngFor='let f of searchedItems' class="col-lg-3 col-md-4 col-sm-6 col-xs-12" style="width: 17rem;">

<div class="card mt-4 " >

  <img [src]='f.Img'  class="card-img-top" [width]=imgWidth1 [height]=imgHeight1 >

  <div class="card-body">

    <h5 class="card-title">{{f.Name}}</h5>

    <p class="card-text">Cost:{{f.Cost|currency:"INR"}}<br>Rating:{{f.Rating}}

    </p>

    <div class="content">

       <button type="button" class="btn1" (click)="inc(f)"><i class="fa fa-plus"></i></button>

       <span>{{f.qnt}}</span>

       <button type="button" class="btn1" (click)="dec(f)"><i class="fa fa-minus"></i></button>

      </div>

      <a  class="btn btn-primary"  (click)="add(f)">Add to Cart</a>

    </div>

  </div>

</div>

</div>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CereComponent } from './cere.component';

describe('CereComponent', () => {

  let component: CereComponent;

  let fixture: ComponentFixture<CereComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CereComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(CereComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { efood } from '../food-items/food-items';

import { foodservice } from 'src/app/food-items/food-items.service';

@Component({

  selector: 'app-nonveg',

  templateUrl: './nonveg.component.html',

  styleUrls: ['./nonveg.component.css']

})

export class NonvegComponent implements OnInit {

  \_searchterm:string;

  imgWidth1="600";

  imgHeight1="300";

  imgRadius="15";

  searchedItems:any[]=[];

    get searchterm():string{

        return this.\_searchterm;

        }

  set searchterm(value:string){

    this.\_searchterm=value;

    this.searchedItems=this.searchterm?this.searchkitchen(this.searchterm):this.cere;

  }

searchkitchen(searchby:string):any[]{

        searchby=searchby.toLocaleLowerCase();

        return this.cere.filter((Food:any)=>Food.Name.toLocaleLowerCase().indexOf(searchby)!==-1);

    }

    constructor(private foo:foodservice) { }

  ngOnInit(): void {

    this.searchedItems=this.cere;

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  itemsCart:any=[];

  add(f)

{

 let cdataNull=localStorage.getItem('cart');

 if(cdataNull==null)

 {

   let sdataget:any=[];

   sdataget.push(f);

   localStorage.setItem('cart',JSON.stringify(sdataget));

 }

 else{

   var id=f.Item;

   let index:number=-1;

   this.itemsCart=JSON.parse(localStorage.getItem('cart'));

   for(let i=0;i<this.itemsCart.length;i++)

   {

     if(parseInt(id) == parseInt(this.itemsCart[i].Item))

     {

         this.itemsCart[i].qnt=f.qnt;

         index=i;

         break;

     }

   }

if(index==-1)

{

  this.itemsCart.push(f);

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

else{

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

 }

 this.cartnumberf();

}

cartNumber:number=0;

  cartnumberf()

  {

      var cartValue=JSON.parse(localStorage.getItem('cart'));

      this.cartNumber= cartValue.length;

      this.foo.cartSubject.next( this.cartNumber);

  }

  cere:efood[]=[

    {

    "Item":1,

  "Name":"Chicken Biriyani",

  "Cost":299,

  "Rating":5.0,

  "qnt":1,

  "Img":"assets/Nv1.png"

  },

  {

    "Item":2,

    "Name":"Chicken Tikka",

    "Cost":350,

    "Rating":4.8,

    "qnt":1,

    "Img":"assets/NV2.png"

    },

    {

      "Item":3,

      "Name":"Chicken Curry",

      "Cost":75,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/NV3.jpg"

      }

      ]

}

import {Injectable} from "@angular/core";

import { efood } from '../food-items/food-items';

import {Subject} from 'rxjs';

@Injectable({

    providedIn:'root'

})

export class cereservice{

  constructor(){}

  // baseurl=global.apiBaseUrl;

   cartSubject=new Subject<any>();

   //openLoginService=new Subject<any>();

 private cere:efood[]=[{

    "Item":1,

  "Name":"Chicken Biriyani",

  "Cost":299,

  "Rating":5.0,

  "qnt":1,

  "Img":"assets/Nv1.png"

  },

  {

    "Item":2,

    "Name":"Chicken Tikka",

    "Cost":350,

    "Rating":4.8,

    "qnt":1,

    "Img":"assets/NV2.png"

    },

    {

      "Item":3,

      "Name":"Chicken Curry",

      "Cost":75,

      "Rating":5.0,

      "qnt":1,

      "Img":"assets/NV3.jpg"

      }

      ]

      getcere():efood[]

      {

         return this.cere;

      }

      getf(id:number):efood{

       const f1=this.cere.find(f1=>f1.Item===id)

       return f1;

      }

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:25px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

.container

{

    color:black;

    font-style: verdana;

    font-size: 20px;

    padding: 10px;

}

* **Generating sfooditems components**

<app-nheader></app-nheader>

<div class="container">

<div class="d-grid gap-2 d-md-flex ">

  &nbsp;&nbsp;&nbsp;

<a [routerLink]="['/fruit']" class="nav" role="button">Veg</a>&nbsp;&nbsp;

<a [routerLink]="['/cere']" class="nav" role="button">Nonveg</a>&nbsp;&nbsp;

<a [routerLink]="['/nut']" class="nav" role="button">Snaksjuice</a><a></a>&nbsp;&nbsp;

</div>

</div>

<div class="container" style="margin-top:20px;">

  <input class="form-control me-3" type="search" placeholder="Search here" [(ngModel)]="searchterm" aria-label="Search" color="green">

</div>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<div class="container" style="margin-top:20px;">

<div class="row card-group">

<div \*ngFor='let f of searchedItems' class="col-lg-3 col-md-4 col-sm-6 col-xs-12" style="width: 17rem;">

<div class="card mt-4 " >

  <img [src]='f.Img'  class="card-img-top" [width]=imgWidth1 [height]=imgHeight1 >

  <div class="card-body">

    <h5 class="card-title">{{f.Name}}</h5>

    <p class="card-text">Cost:{{f.Cost|currency:"INR"}}<br>Rating:{{f.Rating}}

    </p>

    <div class="content">

       <button type="button" class="btn1" (click)="inc(f)"><i class="fa fa-plus"></i></button>

       <span>{{f.qnt}}</span>

       <button type="button" class="btn1" (click)="dec(f)"><i class="fa fa-minus"></i></button>

      </div>

      <a  class="btn btn-primary"  (click)="add(f)">Add to Cart</a>

    </div>

  </div>

</div>

</div>

<app-footer>

</app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { SfooditemsComponent } from './sfooditems.component';

describe('SfooditemsComponent', () => {

  let component: SfooditemsComponent;

  let fixture: ComponentFixture<SfooditemsComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ SfooditemsComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(SfooditemsComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core'

import {efood} from 'src/app/food-items/food-items';

import { foodservice } from 'src/app/food-items/food-items.service';;

@Component({

  selector: 'app-sfooditems',

  templateUrl: './sfooditems.component.html',

  styleUrls: ['./sfooditems.component.css']

})

export class SfooditemsComponent implements OnInit {

  title = 'angular-text-search-highlight';

  \_searchterm:string;

  imgWidth1="600";

  imgHeight1="300";

  imgRadius="15";

  searchedItems:any[]=[];

    get searchterm():string{

        return this.\_searchterm;

        }

  set searchterm(value:string){

    this.\_searchterm=value;

    this.searchedItems=this.searchterm?this.searchkitchen(this.searchterm):this.Food;

  }

searchkitchen(searchby:string):any[]{

        searchby=searchby.toLocaleLowerCase();

        return this.Food.filter((Food:any)=>Food.Name.toLocaleLowerCase().indexOf(searchby)!==-1);

    }

    constructor(private foo:foodservice) { }

  ngOnInit(): void {

    this.searchedItems=this.Food;

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  itemsCart:any=[];

  add(f)

{

 let cdataNull=localStorage.getItem('cart');

 if(cdataNull==null)

 {

   let sdataget:any=[];

   sdataget.push(f);

   localStorage.setItem('cart',JSON.stringify(sdataget));

 }

 else{

   var id=f.Item;

   let index:number=-1;

   this.itemsCart=JSON.parse(localStorage.getItem('cart'));

   for(let i=0;i<this.itemsCart.length;i++)

   {

     if(parseInt(id) == parseInt(this.itemsCart[i].Item))

     {

         this.itemsCart[i].qnt=f.qnt;

         index=i;

         break;

     }

   }

if(index==-1)

{

  this.itemsCart.push(f);

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

else{

  localStorage.setItem('cart',JSON.stringify(this.itemsCart));

}

 }

 this.cartnumberf();

}

cartNumber:number=0;

  cartnumberf()

  {

      var cartValue=JSON.parse(localStorage.getItem('cart'));

      this.cartNumber= cartValue.length;

      this.foo.cartSubject.next( this.cartNumber);

  }

  Food:efood[]=[

    {

     "Item":1,

    "Name":"Chicken Biriyani",

    "Cost":299,

    "Rating":5.0,

    "qnt":1,

    "Img":"assets/Nv1.png"

    },

    {

      "Item":2,

      "Name":"Chicken Tikka",

      "Cost":350,

      "Rating":4.8,

      "qnt":1,

      "Img":"assets/NV2.png"

      },

      {

        "Item":3,

        "Name":"Chicken Curry",

        "Cost":75,

        "Rating":5.0,

        "qnt":1,

        "Img":"assets/NV3.jpg"

        },

        {

          "Item":4,

          "Name":"Veg Biriyani",

          "Cost":100,

          "Rating":5.0,

          "qnt":1,

          "Img":"assets/V1.jpg"

          },

          {

            "Item":5,

            "Name":"Veg Burger",

            "Cost":150,

            "Rating":5.0,

            "qnt":1,

            "Img":"assets/V2.png"

            },

            {

              "Item":6,

              "Name":"Masala Dosa",

              "Cost":250,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/V3.jpg"

              },

               {

              "Item":7,

              "Name":"Samosa",

              "Cost":20,

              "Rating":5.0,

              "qnt":1,

              "Img":"assets/S1.jpg"

              },

              {

                "Item":8,

                "Name":"Chips",

                "Cost":15,

                "Rating":5.0,

                "qnt":1,

                "Img":"assets/S2.jpg"

                },

                {

                  "Item":9,

                  "Name":"French Fries",

                  "Cost":120,

                  "Rating":5.0,

                  "qnt":1,

                  "Img":"assets/S3.webp"

                  },

                  {

                    "Item":10,

                    "Name":"Orage Juice",

                    "Cost":12,

                    "Rating":5.0,

                    "qnt":1,

                    "Img":"assets/J1.jpg"

                    },

                    {

                      "Item":11,

                      "Name":"Apple Juice",

                      "Cost":15,

                      "Rating":5.0,

                      "qnt":1,

                      "Img":"assets/J2.jpg"

                      },

                      {

                        "Item":12,

                        "Name":"Pomegranate Juice",

                        "Cost":15,

                        "Rating":5.0,

                        "qnt":1,

                        "Img":"assets/J3.jpg"

                        }

  ]

}

h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 50px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:25px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 1000px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

p{

    font-style: verdana;

    font-size: 20px;

}

.nav-link{

    font-style: verdana;

    font-size: 20px;

}

.container

{

    color:black;

    font-style: verdana;

    font-size: 20px;

    padding: 10px;

}

**Generating buy components**

<app-nheader></app-nheader>

<section class="cart\_section">

    <div class="container">

    <h2>Cart</h2>

    <div \*ngIf="getCartDetails">

    <div class="table-responsive">

        <table>

            <tr>

                <th>Name</th>

                <th>Price</th>

                <th>Qunatity</th>

                <th>Sub Total</th>

                <th>Remove Item</th>

              </tr>

              <tr \*ngFor="let c of getCartDetails">

                <td>{{c.Name}}</td>

                <td>{{c.Cost| currency:"INR"}}</td>

                <td>

                  <div class="content">

                    <button type="button" class="btn1" (click)="incq(c.Item,c.qnt)"><i class="fa fa-plus"></i></button>

                     <span>{{c.qnt}}</span>

                    <button type="button" class="btn1" (click)="decq(c.Item,c.qnt)"><i class="fa fa-minus"></i></button>

                  </div></td>

                <td>{{c.Cost\*c.qnt|currency:"INR"}}</td>

                <td><button type="button" class="btn btn-danger" (click)="singledel(c.Item)"><i class="fa fa-trash"></i></button></td>

              </tr>

        </table>

    <div class="container">

        <h2>Total:{{total|currency:"INR"}}</h2>

    <div class="col-md-14 text-right">

      <button type="button" class="btn btn-primary" [routerLink]="['/pay']" (click)="removeall()">CheckOut</button>&nbsp;

    <button type="button" class="btn btn-danger" (click)="removeall()">Removeall</button>

    </div>

  </div>

</div>

    </div>

</div>

</section>

<br>

<br>

<br>

<br>

<br>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { BuyComponent } from './buy.component';

describe('BuyComponent', () => {

  let component: BuyComponent;

  let fixture: ComponentFixture<BuyComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ BuyComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(BuyComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Injectable } from '@angular/core';

import { Component, OnInit } from '@angular/core';

import {ActivatedRoute, Router} from "@angular/router";

//import { parse } from 'node:path';

import { efood } from '../food-items/food-items';

import { foodservice } from '../food-items/food-items.service';

@Component({

  selector: 'app-buy',

  templateUrl: './buy.component.html',

  styleUrls: ['./buy.component.css']

})

@Injectable()

export class BuyComponent implements OnInit {

 pgtitle="here are the list of items you selected";

 food:efood;

  constructor(private route:Router,private aroute:ActivatedRoute,private fservice:foodservice) { }

  ngOnInit(): void {

    let id=+this.aroute.snapshot.paramMap.get('id');

    this.pgtitle += `:${id}`;

    this.food = this.fservice.getf(id);

    this.cartdetails();

    this.loadcart();

  }

  incq(Item,qnt)

  {

    for(let i=0;i<this.getCartDetails.length;i++)

    {

      if(this.getCartDetails[i].Item===Item){

        if(qnt!=5)

        this.getCartDetails[i].qnt=parseInt(qnt)+1;

      }

    }

    localStorage.setItem('cart',JSON.stringify(this.getCartDetails));

    this.loadcart();

  }

  decq(Item,qnt)

  {

    for(let i=0;i<this.getCartDetails.length;i++)

    {

      if(this.getCartDetails[i].Item===Item){

        if(qnt!=1)

        this.getCartDetails[i].qnt=parseInt(qnt)-1;

      }

    }

    localStorage.setItem('cart',JSON.stringify(this.getCartDetails));

    this.loadcart();

  }

  inc(f)

  {

    if(f.qnt!=5)

    {

    f.qnt=f.qnt+1;

  }

  }

  dec(f)

  {

    if(f.qnt!=1)

    {

    f.qnt=f.qnt-1;

    }

  }

  removeall()

 {

   localStorage.removeItem('cart');

   this.getCartDetails=[];

   this.total=0;

   this.cartNumber=0;

   this.fservice.cartSubject.next( this.cartNumber);

 }

 singledel(c)

 {

   console.log(c);

   if(localStorage.getItem('cart')){

     this.getCartDetails=JSON.parse(localStorage.getItem('cart'));

     for(let i=0;i<this.getCartDetails.length;i++)

     {

       if(this.getCartDetails[i].Item===c)

       {

         this.getCartDetails.splice(i,1);

         localStorage.setItem('cart',JSON.stringify(this.getCartDetails));

         this.loadcart();

         this.cartnumberf();

       }

     }

   }

 }

 cartNumber:number=0;

 cartnumberf()

 {

     var cartValue=JSON.parse(localStorage.getItem('cart'));

     this.cartNumber= cartValue.length;

     this.fservice.cartSubject.next( this.cartNumber);

 }

  total:number=0;

  loadcart()

  {

    if(localStorage.getItem('cart')){

      this.getCartDetails=JSON.parse(localStorage.getItem('cart'));

      this.total=this.getCartDetails.reduce(function(acc,val){

      return acc+(val.Cost\*val.qnt);

      },0)

    }

  }

  getCartDetails:any=[];

  cartdetails()

  {

    if(localStorage.getItem('cart')){

    this.getCartDetails=JSON.parse(localStorage.getItem('cart'));

    console.log(this.getCartDetails);

    }

  }

 goback()

 {

this.route.navigate(['/Fooditems']);

 }

}

.cart\_section

{

    padding:30px

}

table {

    font-family: arial, sans-serif;

    border-collapse: collapse;

    width: 100%;

  }

  td, th {

    border: 1px solid #dddddd;

    text-align: left;

    padding: 8px;

  }

  tr:nth-child(even) {

    background-color: #dddddd;

  }

  h3

{

    font-family:tangerine;

    color:goldenrod;

    font-size: 25px;

}

h5{

    font-family: tangerine;

    color:indianred;

    font-size:50px;

}

.center

{

    display:block;

    margin-left: auto;

    margin-right:auto;

    width: 100px;

}

.cart\_icon

{

    position: relative;

}

.cart\_icon i{

    font-size: 24px;

}

.cart\_value

{

    position: absolute;

    top:-16px;

    right: -32px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 16px;

}

.content

{

    margin-top:10px;

    margin-left: auto;

    margin-right:auto;

}

.btn1

{

    background:whitesmoke;

    border-radius:50%;

    font-size:16px;

    color:black;

    margin:0 8px;

}

.removeall

{

    padding:4px 10px;

    margin-left: 10px;

    margin-right: 5px;

    color:burlywood

}

* **Generating pay components**

<br>

<div class="container">

    <p>Please Enter your card Details for purchase</p>

<form>

    <div class="form-group">

      <label for="exampleInputEmail1">CardNumber</label>

      <input type="text" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter cardnumber">

    </div>

    <div class="form-group">

      <label for="exampleInputPassword1">Card holder Name</label>

      <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Name">

    </div>

    <div class="form-group">

        <label for="exampleInputPassword1">CVV</label>

        <input type="text" class="form-control" id="exampleInputPassword1" placeholder="cvv">

      </div>

      <div class="form-group">

        <label for="exampleInputPassword1">Expiry Date</label>

        <input type="text" class="form-control" id="exampleInputPassword1" placeholder="MM/YY">

      </div>

    <button type="submit" class="btn btn-primary" [routerLink]="['/final']">Submit</button>

</form>

</div>

<br>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { PayComponent } from './pay.component';

describe('PayComponent', () => {

  let component: PayComponent;

  let fixture: ComponentFixture<PayComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ PayComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(PayComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-pay',

  templateUrl: './pay.component.html',

  styleUrls: ['./pay.component.css']

})

export class PayComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

p

{

    font-family: verdana;

}

* **Generating final components**

<app-nheader></app-nheader>

<div class="container">

    <div class="cart\_icon"><p> Your Order Placed</p>

    <i class="fa fa-check-circle" style="color: green;"></i>

</div><br>

</div>

<br>

<br>

<br>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

<app-footer></app-footer>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { FinalComponent } from './final.component';

describe('FinalComponent', () => {

  let component: FinalComponent;

  let fixture: ComponentFixture<FinalComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ FinalComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(FinalComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-final',

  templateUrl: './final.component.html',

  styleUrls: ['./final.component.css']

})

export class FinalComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

.cart\_icon i{

    position: absolute;

    top:90px;

    right: 600px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 100px;

}

p{

    font-style: verdana;

}

.cart\_icon p{

    text-align: left;

    font-weight: 400;

    font-size: 80px;

    font-family: 'Times New Roman', Times, serif;

}

* **Generating AddNonveg components**

<div align="right">

    <a [routerLink]="['../home']" class="btn btn-info" style="margin:12px;">Logout</a>

    </div>

<div class='btn'>

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

    <a [routerLink]="['../adpwd']" class="btn btn-info" style="margin:12px;">Change password</a>

    <a [routerLink]="['../addcere']" class="btn btn-info" style="margin:12px;">To view and change Nonveg items</a>

    <a [routerLink]="['../addfruits']" class="btn btn-info" style="margin:12px;">To view and change veg items</a>

    <a [routerLink]="['../addnuts']" class="btn btn-info" style="margin:12px;">To view and change snaksjuice</a>

</div>

<div class='btn'>

    <a [routerLink]="['../addic']" class="btn btn-info" style="margin:12px;">Add</a>

</div>

<div class='table-responsive'>

    <table class='table'>

        <thead>

            <tr class="a">

                <th>fooditems</th>

                <th>Name</th>

                <th>Cost</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            <tr \*ngFor='let mov of vtbl'>

                        <img [src]='mov.Img' [width]='imgWidth' [height]='imgHeight'

                            [style.borderRadius.px]='imgRadius'>

                <td>{{mov.Name}}</td>

                <td>{{mov.Cost}}</td>

    <td><button class="btn btn-info" (click)="onDelete(mov.Name)">Delete</button></td>

            </tr>

        </tbody>

    </table>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddcereComponent } from './addcere.component';

describe('AddcereComponent', () => {

  let component: AddcereComponent;

  let fixture: ComponentFixture<AddcereComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AddcereComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AddcereComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { cereservice } from '../nonveg/nonveg.service';

import { efood } from '../food-items/food-items';

@Component({

  selector: 'app-addcere',

  templateUrl: './addnonveg.component.html',

  styleUrls: ['./addnonveg.component.css']

})

export class AddcereComponent implements OnInit {

  imgWidth="100";

  imgHeight="100";

  imgRadius="20";

  vtbl:efood[]=[];

  constructor(private route:Router,private aroute:ActivatedRoute,private utservice:cereservice){}

  ngOnInit(): void {

    this.vtbl=this.utservice.getcere();

      }

      deleteItem(name:string)

      {

        const index=this.vtbl.findIndex(

          item=>item.Name===name

        )

        if(index>=0){

          this.vtbl.splice(index,1);

        }

      }

    onDelete(name:string)

    {

      if(window.confirm("delete"))

      {

        this.deleteItem(name);

        this.route.navigate(['/addcere']);

      }

    }

}

.a

{

    background-color: rgb(240, 214, 235);

}

* **Generating Addveg components**

<div align="right">

    <a [routerLink]="['../home']" class="btn btn-info" style="margin:12px;">Logout</a>

</div>

<div class='btn'>

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

    <a [routerLink]="['../adpwd']" class="btn btn-info" style="margin:12px;">Change password</a>

    <a [routerLink]="['../addcere']" class="btn btn-info" style="margin:12px;">To view and change Nonveg items</a>

    <a [routerLink]="['../addfruits']" class="btn btn-info" style="margin:12px;">To view and change veg items</a>

    <a [routerLink]="['../addnuts']" class="btn btn-info" style="margin:12px;">To view and change snaksjuice</a>

</div>

<div class='btn'>

    <a [routerLink]="['../addi']" class="btn btn-info" style="margin:12px;">Add</a>

</div>

<div class='table-responsive'>

    <table class='table'>

        <thead>

            <tr class="a">

                <th>fooditems</th>

                <th>Name</th>

                <th>Cost</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            <tr \*ngFor='let mov of vtbl'>

                        <img [src]='mov.Img' [width]='imgWidth' [height]='imgHeight'

                            [style.borderRadius.px]='imgRadius'>

                <td>{{mov.Name}}</td>

                <td>{{mov.Cost}}</td>

    <td><button class="btn btn-info" (click)="onDelete(mov.Name)">Delete</button></td>

            </tr>

        </tbody>

    </table>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddfruitsComponent } from './addveg.component';

describe('AddvegComponent', () => {

  let component: AddvegComponent;

  let fixture: ComponentFixture<AddvegComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AddvegComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AddfruitsComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { efood } from '../food-items/food-items';

import { vegservice } from '../veg/veg.service';

@Component({

  selector: 'app-addveg',

  templateUrl: './addveg.component.html',

  styleUrls: ['./addveg.component.css']

})

export class AddvegComponent implements OnInit {

  imgWidth="100";

  imgHeight="100";

  imgRadius="20";

  vtbl:efood[]=[];

  constructor(private route:Router,private aroute:ActivatedRoute,private utservice:fruitservice){}

  ngOnInit(): void {

    this.vtbl=this.utservice.getFruit();

      }

      deleteItem(name:string)

      {

        const index=this.vtbl.findIndex(

          item=>item.Name===name

        )

        if(index>=0){

          this.vtbl.splice(index,1);

        }

      }

    onDelete(name:string)

    {

      if(window.confirm("delete"))

      {

        this.deleteItem(name);

        this.route.navigate(['/addveg']);

      }

    }

}

.a

{

    background-color: rgb(240, 214, 235);

}

* **Generating AddSnaksjuice components**

<div align="right">

    <a [routerLink]="['../home']" class="btn btn-info" style="margin:12px;">Logout</a>

</div>

<div class='btn'>

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

    <a [routerLink]="['../adpwd']" class="btn btn-info" style="margin:12px;">Change password</a>

    <a [routerLink]="['../addcere']" class="btn btn-info" style="margin:12px;">To view and change Nonveg items</a>

    <a [routerLink]="['../addfruits']" class="btn btn-info" style="margin:12px;">To view and change veg items</a>

    <a [routerLink]="['../addnuts']" class="btn btn-info" style="margin:12px;">To view and change snaksjuice</a>

</div>

    <div class='btn'>

        <a [routerLink]="['../addin']" class="btn btn-info" style="margin:12px;">Add</a>

    </div>

    <br><br>

<div class='table-responsive'>

    <table class='table'>

        <thead>

            <tr class="a">

                <th>fooditems</th>

                <th>Name</th>

                <th>Cost</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            <tr \*ngFor='let mov of vtbl'>

                        <img [src]='mov.Img' [width]='imgWidth' [height]='imgHeight'

                            [style.borderRadius.px]='imgRadius'>

                <td>{{mov.Name}}</td>

                <td>{{mov.Cost}}</td>

    <td><button class="btn btn-info" (click)="onDelete(mov.Name)">Delete</button></td>

            </tr>

        </tbody>

    </table>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddnutsComponent } from './addsnaksjuice.component';

describe('AddsnaksjuiceComponent', () => {

  let component: AddsnaksjuiceComponent;

  let fixture: ComponentFixture<AddnutsComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AddsnaksjuiceComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AddsnaksjuiceComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { efood } from '../food-items/food-items';

import { fruitservice } from '../fruit/fruit.service';

import { nutservice } from '../nut/nut.service';

@Component({

  selector: 'app-addnuts',

  templateUrl: './addnuts.component.html',

  styleUrls: ['./addnuts.component.css']

})

export class AddnutsComponent implements OnInit {

  imgWidth="100";

  imgHeight="100";

  imgRadius="20";

  vtbl:efood[]=[];

  constructor(private route:Router,private aroute:ActivatedRoute,private utservice:nutservice){}

  ngOnInit(): void {

    this.vtbl=this.utservice.getnut();

      }

      deleteItem(name:string)

      {

        const index=this.vtbl.findIndex(

          item=>item.Name===name

        )

        if(index>=0){

          this.vtbl.splice(index,1);

        }

      }

    onDelete(name:string)

    {

      if(window.confirm("delete"))

      {

        this.deleteItem(name);

        this.route.navigate(['/addnuts']);

      }

    }

}

.a

{

    background-color: rgb(240, 214, 235);

}

* **Generating Admin components**

<app-header></app-header>

<br>

<br>

<button type="button" class="btn btn-primary" [routerLink]="['/add']">Add</button>

<div class='table-responsive'>

    <table class='table table-bordered'>

        <thead>

            <tr class="bg-info">

                <th>Fruits</th>

                <th>Name</th>

                <th>Cost</th>

                <th>Edit</th>

                <th>Delete</th>

            </tr>

        </thead>

        <tbody>

            <tr \*ngFor='let mov of Food'>

                        <img [src]='mov.Img' [width]='imgWidth' [height]='imgHeight'

                            [style.borderRadius.px]='imgRadius'>

                <td>{{mov.Name}}</td>

                <td>{{mov.Cost}}</td>

                <td><button>Edit</button></td>

    <td><button (click)="onDelete(mov.Name)">Delete</button></td>

            </tr>

        </tbody>

    </table>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdminComponent } from './admin.component';

describe('AdminComponent', () => {

  let component: AdminComponent;

  let fixture: ComponentFixture<AdminComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AdminComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AdminComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-add',

  templateUrl: './add.component.html',

  styleUrls: ['./add.component.css']

})

export class AddComponent implements OnInit {

  constructor() {}

  ngOnInit(): void {

  }

}

body{

    text-align: center;

}

.btn

{

    text-decoration-color: azure;

}

.cart\_icon i{

    position: absolute;

    top:100px;

    right: 600px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 200px;

}

p{

    font-style: verdana;

}

.cart\_icon p{

    position: absolute;

    top:350px;

    right: 600px;;

    background-color: white;

    border:1pxs solid black;

    border-radius: 50%;

    padding:4px 10px;

    font-weight: 800;

    font-size: 50px;

}

* **Generating adpwd components**

<div align="right">

    <a [routerLink]="['../home']" class="btn btn-info" style="margin:12px;">Logout</a>

</div>

&nbsp;&nbsp;&nbsp;<div class='btn'>

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

    <a [routerLink]="['../adpwd']" class="btn btn-info" style="margin:12px;">Change password</a>

    <a [routerLink]="['../addcere']" class="btn btn-info" style="margin:12px;">To view and change Nonveg items</a>

    <a [routerLink]="['../addfruits']" class="btn btn-info" style="margin:12px;">To view and change veg items</a>

    <a [routerLink]="['../addnuts']" class="btn btn-info" style="margin:12px;">To view and change snaksjuice</a>

</div>

<br>

<body>

<div class="card container">

    <div class="card-header heading" align="center" style="background: rgb(105, 255, 138);">

        <p>Please change your password !</p>

    </div>

<div class="card-body lg-4 md-6 sm-12 xs-12" align="center">

    <form novalidate (ngSubmit)="Pass\_change(regForm)" #regForm="ngForm">

        <!-- Password -->

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="Password">Current Password</label>

            <div class="col-md-8">

                <input type="password" required

                id="Password"

                name="pass"

                [(ngModel)]=admin.password

                #pass="ngModel"

                minLength="7"

                [ngClass]="{'is-invalid':(pass.touched||pass.dirty)&&!pass.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=pass.errors?.required>

                        Password is required

                    </span>

                    <span \*ngIf=pass.errors?.minLength>

                        please enter valid password

                    </span>

                </span>

            </div>

        </div>

        <!--New password -->

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="NewPassword">New Password</label>

            <div class="col-md-8">

                <input type="password" required

                id="NewPassword"

                name="pass\_1"

                [(ngModel)]=new\_pass

                #pass\_1="ngModel"

                minlength="7"

                [ngClass]="{'is-invalid':(pass\_1.touched||pass\_1.dirty)&&!pass\_1.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=pass\_1.errors?.required>

                        Password is required

                    </span>

                    <span \*ngIf=pass\_1.errors?.minlength>

                        The Password should be 7 characters long

                    </span>

                </span>

            </div>

        </div>

        <!--Re Enter password -->

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="RNewPassword">Re-enter Password</label>

            <div class="col-md-8">

                <input type="password" required

                id="RNewPassword"

                name="pass\_2"

                [(ngModel)]=r\_new\_pass

                #pass\_2="ngModel"

                minlength="7"

                [ngClass]="{'is-invalid':(pass\_2.touched||pass\_2.dirty)&&!pass\_2.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=pass\_2.errors?.required>

                        Password is required

                    </span>

                    <span \*ngIf=pass\_2.errors?.minlength>

                        The Password should be 7 characters long

                    </span>

                </span>

            </div>

        </div>

        <button class="btn btn-success" type="submit" [disabled]="!regForm.valid">Submit</button>

    </form>

</div>

</div>

</body>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdpwdComponent } from './adpwd.component';

describe('AdpwdComponent', () => {

  let component: AdpwdComponent;

  let fixture: ComponentFixture<AdpwdComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AdpwdComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AdpwdComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { NgForm } from '@angular/forms';

import { Router } from '@angular/router';

import { Admin } from './adpwd';

@Component({

  selector: 'app-adpwd',

  templateUrl: './adpwd.component.html',

  styleUrls: ['./adpwd.component.css']

})

export class AdpwdComponent implements OnInit {

  new\_pass:string;

r\_new\_pass:string;

  constructor(private route:Router) { }

  ngOnInit(): void {

  }

  admin=new Admin();

  Pass\_change(userForm:NgForm){

    if(this.admin.password=='mahathi123')

    {

      if(this.new\_pass==this.r\_new\_pass)

      {

        alert("Admin password changed");

        this.route.navigate(['./add']);

      }

      else{

        alert("Re-enter password correctly");

      }

    }

    else{

      alert("Enter valid current password");

    }

  }

}

body

{

    background-image: url("../../assets/logo1.jpg");

}

p{

    font-style: verdana;

}

* **Generating Addin components**

<div class="card-body lg-4 md-6 sm-12 xs-12" align="center">

    <form novalidate (ngSubmit)="onCereAdd(regForm)" #regForm="ngForm">

        <!-- fruitid -->

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitId">ID</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitId"

                minlength="1"

                name="fruitID"

                [(ngModel)]=kitchen.Item

                #fruitID="ngModel"

                [ngClass]="{'is-invalid':(fruitID.touched||fruitID.dirty)&&!fruitID.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitID.errors?.required>

                        The fruit ID is required

                    </span>

                    <span \*ngIf=fruitID.errors?.minlength>

                        The fruit ID should characters long

                    </span>

                </span>

            </div>

        </div>

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitName">Name</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitName"

                minlength="1"

                name="fruitName"

                [(ngModel)]=kitchen.Name

                #fruitName="ngModel"

                [ngClass]="{'is-invalid':(fruitName.touched||fruitName.dirty)&&!fruitName.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitName.errors?.required>

                        The Name is required

                    </span>

                    <span \*ngIf=fruitName.errors?.minlength>

                        The Name should characters long

                    </span>

                </span>

            </div>

        </div>

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitCost">Cost</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitCost"

                minlength="1"

                name="fruitCost"

                [(ngModel)]=kitchen.Cost

                #fruitCost="ngModel"

                [ngClass]="{'is-invalid':(fruitCost.touched||fruitCost.dirty)&&!fruitCost.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitCost.errors?.required>

                        The cost is required

                    </span>

                    <span \*ngIf=fruitCost.errors?.minlength>

                        The cost should characters long

                    </span>

                </span>

            </div>

        </div>

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitRating">Rating</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitRating"

                minlength="1"

                name="fruitRating"

                [(ngModel)]=kitchen.Rating

                #fruitRating="ngModel"

                [ngClass]="{'is-invalid':(fruitRating.touched||fruitRating.dirty)&&!fruitRating.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitRating.errors?.required>

                        The Rating is required

                    </span>

                    <span \*ngIf=fruitRating.errors?.minlength>

                        The Rating should characters long

                    </span>

                </span>

            </div>

        </div>

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitqnt">quantity</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitqnt"

                minlength="1"

                name="fruitqnt"

                [(ngModel)]=kitchen.qnt

                #fruitqnt="ngModel"

                [ngClass]="{'is-invalid':(fruitqnt.touched||fruitqnt.dirty)&&!fruitqnt.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitqnt.errors?.required>

                        The quantity is required

                    </span>

                    <span \*ngIf=fruitqnt.errors?.minlength>

                        The quantity should characters long

                    </span>

                </span>

            </div>

        </div>

        <div class="form-group row mb-2">

            <label class="col-md-2 col-form-label" for="fruitImg">Image</label>

            <div class="col-md-8">

                <input type="text" required

                id="fruitImg"

                minlength="1"

                name="fruitImg"

                [(ngModel)]=kitchen.Img

                #fruitImg="ngModel"

                [ngClass]="{'is-invalid':(fruitImg.touched||fruitImg.dirty)&&!fruitImg.valid}"/>

                <span class="invalid-feedback">

                    <span \*ngIf=fruitImg.errors?.required>

                        The Rating is required

                    </span>

                    <span \*ngIf=fruitImg.errors?.minlength>

                        The Rating should characters long

                    </span>

                </span>

            </div>

        </div>

        <button class="btn btn-success" type="submit" [disabled]="!regForm.valid">submit</button>

    </form>

</div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddinComponent } from './addin.component';

describe('AddinComponent', () => {

  let component: AddinComponent;

  let fixture: ComponentFixture<AddinComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AddinComponent ]

    })

    .compileComponents();

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(AddinComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

import { Component, OnInit } from '@angular/core';

import { NgForm } from '@angular/forms';

import { Router } from '@angular/router';

import { efood } from '../food-items/food-items';

import { nutservice } from '../nut/nut.service';

import { Kitchens2 } from './addin';

@Component({

  selector: 'app-addin',

  templateUrl: './addin.component.html',

  styleUrls: ['./addin.component.css']

})

export class AddinComponent implements OnInit {

  Item:number;

    Name:string;

    Cost:number;

    Rating:number;

    qnt:number;

    Img:string;

    kitchens1:efood[]=[];

    kitchen=new Kitchens2();

  constructor(private frtservice:nutservice, private route:Router) { }

  ngOnInit(): void {

    this.kitchens1=this.frtservice.getnut();

  }

  save(userForm:NgForm){

    alert("Form Submitted"+this.kitchen.Name);

  }

  onCereAdd(userForm:NgForm){

    let a={

      Item:+this.kitchen.Item,

      Name:this.kitchen.Name,

      Cost:+this.kitchen.Cost,

      Rating:+this.kitchen.Rating,

      qnt:+this.kitchen.qnt,

      Img:this.kitchen.Img,

    }

    this.kitchens1.push(a);

    alert("Item Added");

    this.route.navigate(['./add']);

  }

}

export class Kitchen2{

    Item:number;

    Name:string;

    Cost:number;

    Rating:number;

    qnt:number;

    Img:string;

}

export class Kitchens2{

    constructor(public Item="", public Name="",public Cost="", public Rating="",public qnt="",public Img=""){

    }

}

Back-End Implementaion

Pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.4.3</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>SpringBootEmp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringBootEmp</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-xml</artifactId>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

<finalName>my-movieplan</finalName>

</build>

</project>

**Controller**

package com.foodbox.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.service.IAdminService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/admin")

public class AdminController {

Logger logger = LoggerFactory.getLogger(AdminController.class);

@Autowired

IAdminService adminService;

@PostMapping("/registeradmin/{username}/{password}")

public HttpStatus registerAdmin(@PathVariable String username, @PathVariable String password) throws Exception {

logger.info("-------------------Admin created-----------------");

adminService.registerAdmin(username, password);

return HttpStatus.CREATED;

}

}

package com.foodbox.controller;

import java.time.LocalDate;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.format.annotation.DateTimeFormat;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.AccessForbiddenException;

import com.foodbox.exception.BuyNotFoundException;

import com.foodbox.exception.VegNotFoundException;

import com.foodbox.pojo.Buy;

import com.foodbox.pojo.Veg;

import com.foodbox.service.IBuyService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/buy")

public class BuyController {

Logger logger = LoggerFactory.getLogger(BuyController.class);

@Autowired

private IBuyService bookingService;

@Autowired

LoginController loginController;

@PostMapping(value = "/insert", consumes = MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<Buy> additembuy(@RequestBody Buy t,

@RequestParam(required = false) Integer customerId,@RequestParam(required = false) Integer showId)

throws AccessForbiddenException, BuyNotFoundException {

return ResponseEntity.ok(buyService.addBuy(t, customerId,showId));

}

@GetMapping("/findall")

public ResponseEntity<List<Buy>> viewAllBookings() throws AccessForbiddenException, BuyNotFoundException {

logger.info("-------List Of fooditems Fetched Successfully---------");

return ResponseEntity.ok(bookingService.viewBookingList());

}

@PutMapping(value = "/update", consumes = MediaType.APPLICATION\_JSON\_VALUE)

public Buy updateTicketBooking(@RequestBody Buy t)

throws BuyNotFoundException, AccessForbiddenException {

logger.info("-------fooditems Updated Successfully---------");

return buyService.updateBuy(t);

}

@DeleteMapping("buy/{buyId}")

public Buy deleteTicketBookingById(@PathVariable int bookingId)

throws BuyNotFoundException, AccessForbiddenException {

logger.info("-------Buying With fooditems Id " + buyId + " Deleted Successfully---------");

return buyService.cancelBooking(buyId);

}

@GetMapping("/buy/{buyId}")

public ResponseEntity<Buy> viewBooking(@PathVariable int bookingId)

throws BuyNotFoundException {

ResponseEntity<Buy> response = null;

try {

Buy buy = buyService.viewBuy(buyId);

response = new ResponseEntity<>(booking, HttpStatus.OK);

logger.info("------fooditem Found---------");

} catch (Exception e) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

throw new BuyNotFoundException("fooditem dosen't exist");

}

return response;

}

@GetMapping("/byveg/{vegId}")

public ResponseEntity<List<Buy>> viewvegByvegId(@PathVariable int vegId)

throws AccessForbiddenException, BuyNotFoundException {

logger.info("-------orderss With vegId " +vegId + " Fetched Successfully---------");

return ResponseEntity.ok(bookingService.showAllBookings(movieId));

}

@GetMapping("/byDate/{date}")

public ResponseEntity<List<Buy>> viewvegByLocalDate(

@RequestParam("buyDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date)

throws AccessForbiddenException, BuyNotFoundException {

logger.info("-------Buy With Date " + date + " Fetched Successfully---------");

return ResponseEntity.ok(buyService.showAllBuys(date));

}

@GetMapping("/puy/{buyId}")

public double TotalBookingost(@PathVariable int bookingId)

throws AccessForbiddenException, BuyNotFoundException {

logger.info("-------Total Cost Of orders displayed Successfully---------");

return buyService.calculateTotalCost(buyId);

}

}

package com.foodbox.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.AccessForbiddenException;

import com.foodbox.exception.CustomerNotFoundException;

import com.foodbox.exception.FooditemNotFoundException;

import com.foodbox.pojo.Customer;

import com.foodbox.pojo.Fooditms;

import com.foodbox.service.CustomerService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/customer")

public class CustomerController {

Logger logger = LoggerFactory.getLogger(CustomerController.class);

@Autowired

private CustomerService customerService;

@Autowired

LoginController loginController;

@PostMapping("/additems")

public ResponseEntity<String> addCustomer(@RequestBody Customer customer)

throws CustomerNotFoundException, AccessForbiddenException {

ResponseEntity<String> response = null;

if (customer == null) {

logger.error("-------------Please Enter Customer Values--------");

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

customer = customerService.addCustomer(customer);

response = new ResponseEntity<>("Customer is Added", HttpStatus.CREATED);

logger.info("----------------Customer Created------------------");

}

return response;

}

@DeleteMapping("/delete/{customerId}")

public Customer removeCustomer(@PathVariable int customerId)

throws CustomerNotFoundException, AccessForbiddenException {

logger.info("----------------Customer Deleted Successfully--------------");

return customerService.removeCustomer(customerId);

}

@PutMapping("/update")

public ResponseEntity<Customer> updateCustomer(@RequestBody Customer customer)

throws CustomerNotFoundException, AccessForbiddenException {

ResponseEntity<Customer> response = null;

if (customer == null) {

logger.error("Enter Customer Details");

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

customer = customerService.updateCustomer(customer);

response = new ResponseEntity<>(customer, HttpStatus.OK);

logger.info("--------------Customer Updated Successfully-----------------");

}

return response;

}

@GetMapping("/findall")

public List<Customer> viewCustomerList() throws AccessForbiddenException {

logger.info("---------------Customer List-----------------");

return customerService.viewCustomerList();

}

@GetMapping("/view/{customerId}")

public ResponseEntity<Customer> viewACustomer(@PathVariable int customerId) throws CustomerNotFoundException {

ResponseEntity<Customer> response = null;

try {

Customer movie = customerService.viewCustomer(customerId);

response = new ResponseEntity<>(movie, HttpStatus.OK);

logger.info("------fooditems With itemId id " + customerId + " Found---------");

} catch (Exception e) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

throw new CustomerNotFoundException("Customer with " + customerId + " id dosen't exist");

}

return response;

}

}

package com.foodbox.controller;

import java.io.IOException;

import java.time.LocalDate;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.format.annotation.DateTimeFormat;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

import com.foodbox.exception.FooditemNotFoundException;

import com.foodbox.pojo.Fooditms;

import com.foodbox.service.FooditemsService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/fooditems")

public class FooditemsController {

Logger logger = LoggerFactory.getLogger(FooditemsController.class);

@Autowired

private foodService foodService;

@PostMapping("/add")

public ResponseEntity<fooditems> addMovie(@RequestBody food food)

throws FooditemNotFoundException, IOException {

movie = moviesService.addMovie(movie);

logger.info("-------fooditems Added Successfully---------");

return new ResponseEntity<>(movie, HttpStatus.CREATED);

}

@PutMapping("/update")

public ResponseEntity<Fooditms> updateMovie(@RequestBody Fooditms movie)

throws FooditemNotFoundException {

ResponseEntity<Fooditms> response = null;

if (movie == null) {

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

fooditems = fooditemsService.updatefoodtems(food);

response = new ResponseEntity<>(movie, HttpStatus.OK);

logger.info("-------fooditems Updated Successfully---------");

}

return response;

}

@PutMapping("/map")

public ResponseEntity<fooditems> addToShow(@RequestBody fooditems food,@RequestParam(required = false) Integer showId)

throws FooditemNotFoundException {

ResponseEntity<fooditems> response = null;

if (food == null) {

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

food = foodService.addFoodToShow(food,showId);

response = new ResponseEntity<>(food, HttpStatus.OK);

logger.info("-------Movie Updated Successfully---------");

}

return response;

}

@GetMapping("/findall")

public ResponseEntity<List<fooditmes>> viewfooditemsList() throws FoodNotFoundException {

logger.info("-------Movie List Fetched---------");

return ResponseEntity.ok(foodService.viewfooditemsList());

}

@GetMapping("/viewMovie/{movieId}")

public ResponseEntity<Fooditms> viewMovie(@PathVariable int movieId)

throws FooditemNotFoundException {

ResponseEntity<Fooditms> response = null;

try {

Fooditms movie = moviesService.viewMovie(movieId);

response = new ResponseEntity<>(movie, HttpStatus.OK);

logger.info("-------fooditems With fooditem id " + foodId + " Found---------");

} catch (Exception e) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

throw new foodNotFoundException("fooditems with " + foodId + " id dosen't exist");

}

return response;

;

}

@DeleteMapping("/delete/{movieId}")

public ResponseEntity<fooditems> removefooditems(@PathVariable int foodId)

throws foodNotFoundException {

ResponseEntity<Fooditms> response = null;

Fooditems food = Service.viewFood(foodId);

if (food == null) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

} else {

foodService.removeFooditems(foodId);

response = new ResponseEntity<>(food, HttpStatus.OK);

logger.info("-------fooditem With food id " + foodId + " Deleted---------");

}

return response;

}

}

package com.foodbox.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.pojo.Login;

import com.foodbox.service.LoginService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class LoginController {

Logger logger = LoggerFactory.getLogger(LoginController.class);

@Autowired

LoginService logServ;

@PostMapping("/login/{username}/{password}")

public Login loginUser(@PathVariable String username, @PathVariable String password) {

Login login=new Login();

try {

login=logServ.loginWithData(username, password);

} catch (Exception e) {

logger.error("------------------LoginFailed---------------");

return login;

}

logger.info("-----------------Login Successful----------------");

return login;

}

@PostMapping("/Logout")

public HttpStatus logOut() throws Exception {

if (this.loginStatus()) {

logServ.logoutPresentUser();

return HttpStatus.ACCEPTED;

} else {

throw new Exception("User Not yet Lo gged In");

}

}

public boolean loginStatus() {

return logServ.loginStatus();

}

public String getRole() {

return logServ.getRole();

}

}

package com.foodbox.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.AccessForbiddenException;

import com.foodbox.exception.NonvegNotFoundException;

import com.foodbox.pojo.NonVeg;

import com.foodbox.service.ISeatService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/seats")

public class NonVegController {

Logger logger = LoggerFactory.getLogger(NonVegController.class);

@Autowired

private ISeatService seatService;

@Autowired

LoginController loginController;

@PostMapping("/addNonVeg")

public ResponseEntity<NonVeg> addANonVeg(@RequestBody NonVeg nonveg)

throws AccessForbiddenException, NonVegNotFoundException {

nonveg = nonvegService.addSeat(seat);

logger.info("-------NonVeg Added Successfully---------");

return new ResponseEntity<>(nonveg, HttpStatus.CREATED);

}

@GetMapping("/findall")

public ResponseEntity<List<NonVeg>> viewNonVegList() throws AccessForbiddenException, NonVegNotFoundException {

logger.info("-------List of NonVeg Fetched Successfully---------");

return ResponseEntity.ok(nonvegService.viewNonVegList());

}

@PutMapping("/update")

public ResponseEntity<NonVeg> updateNonVeg(@RequestBody NonVeg nonveg)

throws AccessForbiddenException, NonvegNotFoundException {

ResponseEntity<NonVeg> response = null;

if (nonveg == null) {

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

nonveg = nonvegService.updateNonVeg(nonveg);

response = new ResponseEntity<>(nonveg, HttpStatus.OK);

logger.info("-------NonVeg Updated Successfully---------");

}

return response;

}

@PutMapping("/buy")

public ResponseEntity<NonVeg> BuyANonVeg(@RequestBody NonVeg nonveg)

throws AccessForbiddenException, NonVegNotFoundException {

nonveg = nonvegService.buyNonVeg(nonveg);

logger.info("-------NonVeg buying Successfull---------");

return new ResponseEntity<>(nonveg, HttpStatus.OK);

}

@PutMapping("/cancel")

public ResponseEntity<NonVeg> CancelASeat(@RequestBody NonVeg nonveg)

throws AccessForbiddenException, NonVegNotFoundException {

nonveg = nonvegService.cancelNonVegBuying(nonveg);

logger.info("-------NonVeg Cancellation Successfull---------");

return new ResponseEntity<>(nonveg, HttpStatus.OK);

}

}

package com.foodbox.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.AccessForbiddenException;

import com.foodbox.exception.PuyNotFoundException;

import com.foodbox.pojo.Puy;

import com.foodbox.service.PuyService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/puy")

public class PuyController {

Logger logger = LoggerFactory.getLogger(PuyController.class);

@Autowired

private PuyService puyService;

@Autowired

LoginController loginController;

@PostMapping("/puy")

public ResponseEntity<Puy> addATicket(@RequestBody Puy puy,@RequestParam(required = false) Integer buyId)

throws AccessForbiddenException, PuyNotFoundException {

Puy = puyService.addPuy(puy,buyId);

logger.info("-------Puy Created Successfully---------");

return new ResponseEntity<>(puy, HttpStatus.CREATED);

}

@GetMapping("/findall")

public ResponseEntity<List<Puy>> viewPuyList() throws AccessForbiddenException, PuyNotFoundException {

return ResponseEntity.ok(puyService.viewPuyList());

}

@GetMapping("/{PuyId}")

public Puy findATicket(@PathVariable int puyId) throws PuyNotFoundException, AccessForbiddenException {

Puy t = null;

try {

t = puyService.findTicket(puyId);

logger.info("-------Puy with puyId " + puyId + " Foound Successfully---------");

} catch (Exception e) {

throw new PuyNotFoundException("Invalid Puyment ID");

}

return t;

}

}

package com.foodbox.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.ShowNotFoundException;

import com.foodbox.pojo.Show;

import com.foodbox.service.ShowService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/snaksjuice")

public class SnaksJuiceController {

Logger logger = LoggerFactory.getLogger(SnaksJuiceController.class);

@Autowired

private SnakJuiceService snakjuiceService;

@PostMapping("/addSnakJuic")

public ResponseEntity<SnakJuic> addSnakJuic(@RequestBody SnakJuic snakjuic, @RequestParam(required = false) ,

@RequestParam(required = false) Integer SnakJuicId) {

SnakJuicService.addSnakJuic(snakjuic, SnakJuicId);

logger.info("-------SnakJuic Added Succesfully--------");

return new ResponseEntity<>(SnakJuic, HttpStatus.CREATED);

}

@DeleteMapping("/delete/{SnakJuicId}")

public ResponseEntity<SnakJuic> removeSnakJuic(@PathVariable int showId) {

ResponseEntity<SnakJuic> response = null;

SnakJuic snakjuic = SnakJuicService.viewSnakJuic(SnakJuicId);

if (snakjuic == null) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

} else {

snakjuicService.removeSnakJuic(snakjuicId);

response = new ResponseEntity<>(snakjuic, HttpStatus.OK);

logger.info("-------SnakJuic with SnakJuicId " + snakjuicId + " Deleted Successfully---------");

}

return response;

}

@PutMapping("/update")

public ResponseEntity<Show> updateShow(@RequestBody Show show, @RequestParam(required = false) Integer theatreId,

@RequestParam(required = false) Integer screenId) {

ResponseEntity<Show> response = null;

if (show == null) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

} else {

showService.updateShow(show, theatreId, screenId);

response = new ResponseEntity<>(show, HttpStatus.OK);

logger.info("-------Show Updated Successfully---------");

}

return response;

}

@GetMapping("/view/{SnakJuicId}")

public ResponseEntity<SnakJuic> viewShow(@PathVariable int SnakJuicId)

throws SnakJuicNotFoundException {

ResponseEntity<SnakJuic> response = null;

try {

SnakJuic snakjuic = SnakJuicService.viewSnakJuic(snakjuicId);

response = new ResponseEntity<>(snakjuic, HttpStatus.OK);

logger.info("-------SnakJuic with SnakJuicId " + snakjuicId + " Found Successfully---------");

} catch (Exception e) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

throw new SnakJuicNotFoundException("SnakJuic with " + snakjuicId + " id dosen't exist");

}

return response;

}

}

package com.foodbox.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.foodbox.exception.ScreenNotFoundException;

import com.foodbox.pojo.Screen;

import com.foodbox.pojo.Theatre;

import com.foodbox.service.ScreenService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/veg")

public class VegitemController {

Logger logger = LoggerFactory.getLogger(ScreenController.class);

@Autowired

private VegService vegService;

@PostMapping("/add")

public ResponseEntity<Vegitem> addAVegitem(@RequestBody Veg veg,

@RequestParam(required = false) )

throws VegNotFoundException {

logger.info("-------Vegitem Successfully added into cart " + vegitemId + "---------");

return ResponseEntity.ok(VegService.addVegitem(veg, vegitemId));

}

@GetMapping("/findall")

public ResponseEntity<List<Vegitem> viewScreenList() throws VegitemNotFoundException {

logger.info("-------List Of veg items Fetched Successfully---------");

return ResponseEntity.ok(vegService.viewCartList());

}

@GetMapping("/veg/{vegId}")

public ResponseEntity<Veg> getVegById(@PathVariable int vegId) throws VegNotFoundException {

ResponseEntity<Veg> response = null;

try {

Veg veg = vegitemService.getVeg(vegitemId);

response = new ResponseEntity<>(veg, HttpStatus.OK);

} catch (Exception e) {

response = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

return response;

}

@PutMapping("/update")

public ResponseEntity<Vegitem> updateVegitem(@RequestBody Vegitem s, @RequestParam(required = false) )

throws VegNotFoundException {

ResponseEntity<Vegitem> response = null;

if (s == null) {

response = new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

s = vegitemService.updateScreen(s, VegId);

response = new ResponseEntity<>(s, HttpStatus.OK);

logger.info("-------vegitem Updated Successfully---------");

}

return response;

}

}

Ex

**package** com.foodbox.exception;

**public** **class** AccessForbiddenException **extends** Exception {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** AccessForbiddenException(String message) {

**super**(message);

}

}

v**package** com.foodbox.exception;

**public** **class** BuyNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** BuyNotFoundException() {

}

**public** BuyNotFoundException(String message) {

**super**(message);

}

}

**package** com.foodbox.exception;

**public** **class** CustomerNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** CustomerNotFoundException() {

}

**public** CustomerNotFoundException(String message) {

**super**(message);

}

}

**package** com.foodbox.exception;

**public** **class** FooditemNotFoundException **extends** Exception {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** FooditemNotFoundException() {

}

**public** FooditemNotFoundException(String message) {

**super**(message);

}

}

**package** com.foodbox.exception;

**public** **class** NonvegNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** NonvegNotFoundException() {

}

**public** NonvegNotFoundException(String message) {

**super**(message);

}

}

**package** com.foodbox.exception;

**public** **class** PuyNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** PuyNotFoundException(String msg) {

**super**(msg);

}

}

**package** com.foodbox.exception;

**public** **class** UserNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** UserNotFoundException(String message) {

**super**(message);

}

}

**package** com.foodbox.exception;

**public** **class** VegNotFoundException **extends** Exception {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** VegNotFoundException(String msg) {

**super**(msg);

}

}

**Repository**

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.foodbox.pojo.Buy;

@Repository

public interface BuyRepository extends JpaRepository<Buy, Integer> {

}

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.foodbox.pojo.Customer;

@Repository

public interface CustomerRepository extends JpaRepository<Customer, Integer> {

boolean existsByMobileNumber(String mobileNumber);

boolean existsByEmail(String email);

}

package com.foodbox.repository;

import java.time.LocalDate;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.foodbox.pojo.Fooditms;

@Repository

public interface Fooditems extends JpaRepository<Fooditms, Integer> {

}

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.foodbox.pojo.NonVeg;

@Repository

public interface NonVegRepository extends JpaRepository<NonVeg, Integer> {

}

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.foodbox.pojo.Puy;

public interface PuyRepository extends JpaRepository<Puy, Integer> {

}

package com.foodbox.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import com.foodbox.pojo.SnaksJuice;

public interface SnaksJuiceRepository extends JpaRepository<SnaksJuice, Integer> {

}

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.foodbox.pojo.User;

public interface UserRepository extends JpaRepository<User, Integer> {

}

package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.foodbox.pojo.Veg;

public interface VegRepository extends JpaRepository<Veg, Integer> {

}

**Pojo**

package com.foodbox.pojo;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int customerId;

private String customerName;

private String address;

private String mobileNumber;

private String email;

private String password;

// @JsonIgnore

// @OneToOne

// private User user;

public Customer() {

}

public Customer(String customerName, String address, String mobileNumber, String email, String password) {

super();

this.customerName = customerName;

this.address = address;

this.mobileNumber = mobileNumber;

this.email = email;

this.password = password;

}

public int getCustomerId() {

return customerId;

}

public void setCustomerId(int customerId) {

this.customerId = customerId;

}

public String getCustomerName() {

return customerName;

}

public void setCustomerName(String customerName) {

this.customerName = customerName;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getMobileNumber() {

return mobileNumber;

}

public void setMobileNumber(String mobileNumber) {

this.mobileNumber = mobileNumber;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

package com.foodbox.pojo;

import java.sql.Blob;

import java.time.LocalDate;

import javax.persistence.\*;

import org.hibernate.annotations.DynamicUpdate;

import com.fasterxml.jackson.annotation.JsonFormat;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;

import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateDeserializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateSerializer;

@Entity

@Table(name = "movie")

@DynamicUpdate

public class Fooditms {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int fooditemId;

private String fooditemName;

private String fooditemOption;

private String fooditemPrice;

private String fooditemCatogiry;

private String fooditemDescription;

private String fooditemRating;

@JsonDeserialize(using = LocalDateDeserializer.class)

@JsonSerialize(using = LocalDateSerializer.class)

@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")

private LocalDate movieDate;

@JsonIgnore

@OneToOne

private SnaksJuice show;

public Fooditms() {

}

public Fooditms(String Name, String catogry, String movieLanguage, String Description,

String Rating, SnaksJuice show) {

super();

this.fooditemName = fooditemName;

this.fooditemOption = fooditemOption;

this.fooditemPrice = fooditemPrice;

this.fooditemCatogiry = fooditemCatogiry;

this.fooditemDescription = fooditemDescription;

this.fooditemRating = fooditemRating;

}

public int getFooditemId() {

return fooditemId;

}

public void setFooditemId(int fooditemId) {

this.fooditemId = fooditemId;

}

public String getFooditemName() {

return fooditemName;

}

public void setFooditemName(String fooditemName) {

this.fooditemName = fooditemName;

}

public String getFooditemOption() {

return fooditemOption;

}

public void setFooditemOption(String fooditemOption) {

this.fooditemOption = fooditemOption;

}

public String getFooditemPrice() {

return fooditemPrice;

}

public void setFooditemPrice(String fooditemPrice) {

this.fooditemPrice = fooditemPrice;

}

public String getFooditemCatogiry() {

return fooditemCatogiry;

}

public void setFooditemCatogiry(String fooditemCatogiry) {

this.fooditemCatogiry = fooditemCatogiry;

}

public String getFooditemDescription() {

return fooditemDescription;

}

public void setFooditemDescription(String fooditemDescription) {

this.fooditemDescription = fooditemDescription;

}

public String getFooditemRating() {

return fooditemRating;

}

public void setFooditemRating(String fooditemRating) {

this.fooditemRating = fooditemRating;

}

}

package com.foodbox.pojo;

import javax.persistence.\*;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

public class NonVeg {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int nonvegId;

private String nonvegName;

private String type;

private double price;

@Enumerated(EnumType.STRING)

private SeatStatus status;

@JsonIgnore

@ManyToOne

private Puy puy;

public NonVeg() {

}

public NonVeg(String nonvegName, String type, double price, Puy puy) {

super();

this.nonvegName = nonvegName;

this.type = type;

this.price = price;

this.puy = puy;

}

public int getNonvegId() {

return nonvegId;

}

public void setNonvegId(int nonvegId) {

this.nonvegId = nonvegId;

}

public String getNonvegName() {

return nonvegName;

}

public void setNonvegName(String nonvegName) {

this.nonvegName = nonvegName;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public Puy getPuy() {

return puy;

}

public void setPuy(Puy puy) {

this.puy = puy;

}

}

package com.foodbox.pojo;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

@Entity

public class Puy {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int ticketId;

private int noOfSeats;

private boolean ticketStatus;

//@OneToMany(mappedBy = "ticket")

@OneToMany

private List<NonVeg> seats;

@OneToOne

private Buy booking;

public Puy() {

}

public Puy(int noOfSeats, boolean ticketStatus, List<NonVeg> seats, Buy booking) {

super();

this.noOfSeats = noOfSeats;

this.ticketStatus = ticketStatus;

this.seats = seats;

this.booking = booking;

}

public int getTicketId() {

return ticketId;

}

public void setTicketId(int ticketId) {

this.ticketId = ticketId;

}

public int getNoOfSeats() {

return noOfSeats;

}

public void setNoOfSeats(int noOfSeats) {

this.noOfSeats = noOfSeats;

}

public boolean isTicketStatus() {

return ticketStatus;

}

public void setTicketStatus(boolean ticketStatus) {

this.ticketStatus = ticketStatus;

}

public List<NonVeg> getSeats() {

return seats;

}

public void setSeats(List<NonVeg> seats) {

this.seats = seats;

}

public Buy getBooking() {

return booking;

}

public void setBooking(Buy booking) {

this.booking = booking;

}

}

package com.foodbox.pojo;

import javax.persistence.\*;

import org.hibernate.annotations.DynamicUpdate;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;

import com.fasterxml.jackson.databind.annotation.JsonSerialize;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateTimeDeserializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateDeserializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateSerializer;

import java.time.\*;

@Entity

@Table(name = "snakjuice")

@DynamicUpdate

public class SnaksJuice {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int snakjuiceId;

@DateTimeFormat(iso = DateTimeFormat.ISO.DATE\_TIME)

@JsonSerialize(using = LocalDateTimeSerializer.class)

@JsonDeserialize(using = LocalDateTimeDeserializer.class)

private int snakjuicid;

private String snakjuiceName;

@OneToOne(mappedBy = "snakjuice")

private Fooditms fooditem;

@JsonIgnore

@ManyToOne

private Veg veg;

@JsonIgnore

@OneToOne

private Buy buy;

private LocalDate showDate;

public SnaksJuice() {

}

public int getSnakjuiceId() {

return snakjuiceId;

}

public void setSnakjuiceId(int snakjuiceId) {

this.snakjuiceId = snakjuiceId;

}

public int getSnakjuicid() {

return snakjuicid;

}

public void setSnakjuicid(int snakjuicid) {

this.snakjuicid = snakjuicid;

}

public String getSnakjuiceName() {

return snakjuiceName;

}

public void setSnakjuiceName(String snakjuiceName) {

this.snakjuiceName = snakjuiceName;

}

public Fooditms getFooditem() {

return fooditem;

}

public void setFooditem(Fooditms fooditem) {

this.fooditem = fooditem;

}

public Veg getVeg() {

return veg;

}

public void setVeg(Veg veg) {

this.veg = veg;

}

public Buy getBuy() {

return buy;

}

public void setBuy(Buy buy) {

this.buy = buy;

}

}

package com.foodbox.pojo;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import org.hibernate.annotations.DynamicUpdate;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

@DynamicUpdate

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int userid;

private String username;

private String password;

private String role;

@OneToOne

private Customer customer;

User() {

}

public User(String username, String password, String role, Customer customer) {

super();

this.username = username;

this.password = password;

this.role = role;

this.customer = customer;

}

public int getUserid() {

return userid;

}

public void setUserid(int userid) {

this.userid = userid;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

}

package com.foodbox.pojo;

import java.util.List;

import javax.persistence.\*;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

public class Veg {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int screenId;

@JsonIgnore

@ManyToOne

@OneToMany(mappedBy = "veg",cascade = CascadeType.ALL)

private List<SnaksJuice> show;

private String vegName;

@Column(name = "rowss")

private int rows;

@Column(name = "columnss")

private int columns;

public Veg() {

}

public Veg( List<Veg> show, String vegName, int rows, int columns) {

super();

this.vegName = vegName;

this.rows = rows;

this.columns = columns;

}

public int getScreenId() {

return screenId;

}

public void setScreenId(int screenId) {

this.screenId = screenId;

}

public String getVegName() {

return vegName;

}

public void setVegName(String vegName) {

this.vegName = vegName;

}

public int getRows() {

return rows;

}

public void setRows(int rows) {

this.rows = rows;

}

public int getColumns() {

return columns;

}

public void setColumns(int columns) {

this.columns = columns;

}

}

**Services**

package com.foodbox.service;

import java.util.List;

import com.foodbox.exception.CustomerNotFoundException;

import com.foodbox.pojo.Customer;

public interface CustomerService {

boolean existsById(Integer userId);

boolean existsByMobileNumber(String mobileNo);

boolean existsByEmail(String email);

public Customer addCustomer(Customer customer) throws CustomerNotFoundException;

public Customer removeCustomer(int customerid);

public Customer updateCustomer(Customer customer) throws CustomerNotFoundException;

public Customer viewCustomer(int customerid);

public List<Customer> viewCustomerList();

}

package com.foodbox.service;

import java.util.List;

import org.springframework.stereotype.Service;

import com.foodbox.exception.CustomerNotFoundException;

import com.foodbox.pojo.Customer;

import com.foodbox.repository.CustomerRepository;

@Service

public class CustomerServiceImpl implements CustomerService {

private CustomerRepository customerRepository;

public CustomerServiceImpl(CustomerRepository customerRepository) {

this.customerRepository = customerRepository;

}

@Override

public Customer addCustomer(Customer customer) throws CustomerNotFoundException {

if (customer != null) {

if (customerRepository.existsById(customer.getCustomerId())) {

throw new CustomerNotFoundException("Customer " + customer.getCustomerId() + " is already Exists");

} else if (customerRepository.existsByMobileNumber(customer.getMobileNumber())) {

throw new CustomerNotFoundException(

"Customer with number " + customer.getMobileNumber() + " is already Exists");

} else if (customerRepository.existsByEmail(customer.getEmail())) {

throw new CustomerNotFoundException(

"Customer with email " + customer.getEmail() + " is already Exists");

} else {

customerRepository.saveAndFlush(customer);

}

}

return customerRepository.getOne(customer.getCustomerId());

}

@Override

public List<Customer> viewCustomerList() {

return customerRepository.findAll();

}

@Override

public Customer removeCustomer(int customerid) {

Customer c = customerRepository.getOne(customerid);

customerRepository.delete(c);

return c;

}

@Override

public Customer updateCustomer(Customer customer) throws CustomerNotFoundException {

/\*

\* Customer cu=customerRepository.getOne(customer.getCustomerId());

\* System.out.println(cu.getCustomerName());

\* System.out.println(cu.getPassword());

\*/

System.out.println(customer.getCustomerId());

customerRepository.saveAndFlush(customer);

return customerRepository.getOne(customer.getCustomerId());

}

@Override

public Customer viewCustomer(int customerid) {

return customerRepository.findById(customerid).get();

}

@Override

public boolean existsById(Integer userId) {

return customerRepository.existsById(userId);

}

@Override

public boolean existsByMobileNumber(String mobileNo) {

// TODO Auto-generated method stub

return customerRepository.existsByMobileNumber(mobileNo);

}

@Override

public boolean existsByEmail(String email) {

// TODO Auto-generated method stub

return customerRepository.existsByEmail(email);

}

}

**package** com.foodbox.service;

**import** org.springframework.stereotype.Service;

@Service

**public** **interface** IAdminService {

**public** **void** registerAdmin(String username, String password) **throws** Exception;

}

package com.foodbox.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cg.mts.validator.InputValidator;

import com.foodbox.exception.UserCreationError;

import com.foodbox.pojo.User;

import com.foodbox.repository.IAdminRepository;

@Service

public class IAdminServiceImpl implements IAdminService {

@Autowired

IAdminRepository adminRepo;

@Autowired

InputValidator validate;

@Override

public void registerAdmin(String username, String password) throws Exception {

if (!validate.usernameValidator(username))

throw new UserCreationError("Check Username !!!!");

if (!validate.passwordValidator(password))

throw new UserCreationError("Cannot register this admin with this password");

User u = new User(username, password, "ADMIN", null);

adminRepo.saveAndFlush(u);

}

}

package com.foodbox.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cg.mts.repoImpl.QueryClass;

import com.foodbox.pojo.Login;

import com.foodbox.pojo.User;

@Service

public class LoginService {

@Autowired

private QueryClass qcp;

private Login logData = new Login();

public Login loginWithData(String username, String password) throws Exception {

User user = qcp.findByUserName(username);

if (!user.getPassword().equals(password))

throw new Exception("Login Data Invalid");

logData.setLoginStatus(true);

logData.setUser(user);

return logData;

}

public Login logoutPresentUser() {

if (logData.isLoginStatus()) {

logData.setLoginStatus(false);

}

return logData;

}

public boolean loginStatus() {

return logData.isLoginStatus();

}

public String getRole() {

return logData.getRole();

}

}

package com.foodbox.service;

import java.util.List;

import com.foodbox.exception.PuyNotFoundException;

import com.foodbox.pojo.Puy;

public interface PuyService {

public Puy addPuy(Puy puy,Integer buyId) throws PuyNotFoundException;

public Puy findPuy(int puyId);

List<Puy> viewPuyList() throws PuyNotFoundException;

}

**package** com.foodbox.service;

**import** java.time.LocalDate;

**import** java.util.List;

**import** com.foodbox.pojo.SnaksJuice;

**public** **interface** SnaksJuiceService {

**public** SnaksJuice addShow(SnaksJuice showInteger vegId);

**public** SnaksJuice updateShow(SnaksJuice show, Integer vegId);

**public** SnaksJuice removeShow(**int** showid);

**public** SnaksJuice viewShow(**int** showid);

**public** List<SnaksJuice> viewAllShows();

}

package com.foodbox.service;

import java.util.List;

import com.foodbox.exception.VegNotFoundException;

import com.foodbox.pojo.Veg;

public interface VegService {

public Veg addVeg(Veg veg, Integer vegid) throws VegNotFoundException;

public List<Veg> viewVegList() throws VegNotFoundException;

public Veg updateVeg(Veg s, Integer VegId);

public Veg viewVeg(int screenId) throws VegNotFoundException;

}